

GE Fanuc Automation

可编程控制产品

CPU 参考手册

GFL-002

通告，警告，和注意
当使用这出版物时

通知

通知用在这出版物中，强调危险电压，电流，温度，或这个设备存在或与设备相连的其他可能引起人身伤害的情况.

如果疏忽，可能引起其他人身伤害或损坏设备，因此使用通告。

警告

警告用于如果不小心，设备就会损坏的情况。

注意: 仅仅关注非常重要的理解和操作设备的信息。

这个文件基于被发行时可利用的信息。当成果已经更精确时，这里包含的信息完全覆盖硬件或软件的所有详细内容或变化，不可能提供安装，运行，或维护中的每个可能的偶然事件。 这里描述的特征并不存在于所有硬件和软件系统中。当后续版本发生变化，GE Fanuc 自动化没有义务通知文件持有者。

关于这里所包含的信息的准确度，完整性，充足性，或有效性，GE Fanuc 自动化不负责代表性或 保证性，明确的，暗指的，或遵从法定的。没有商品材或适用性保证适用。

如下是 GE Fanuc 自动化北美公司的商标。

Alarm Master	Genius	PROMACRO	Series Three
CIMPLICITY	Helpmate	PowerMotion	VersaMax
CIMPLICITY 90–ADS	Logicmaster	PowerTRAC	VersaPoint
CIMSTAR	Modelmaster	Series 90	VersaPro
Field Control	Motion Mate	Series Five	VuMaster
FrameworX	PACSystems	Series One	Workmaster
GEnet	ProLoop	Series Six	

Contents

Contents

Contents

@

A

B

C

G

H

I

J

L

M

N

O

P

R

T

U

V

W

Y

Z

Chapter 1 简介

本手册介绍 PAC 系统 CPU 操作和编程的通用信息，同时也对特定程序的要求作了细节描述。

第一章对 PAC 系统产品作了一般性的介绍，其中包括 PAC 的特点，产品综述和相关文献列表。

第二章介绍 CPU 硬件的特点并作了相关说明

安装过程在 PAC RX7i 和 PAC RX3i 安装手册(编号分别为 GFK2223 和 GFK2314)中作了介绍

第三章介绍了怎样进行 CPU 配置，如何利用软件中配置的参数确定模块的操作特征，并为每个模块分配系统地址。

第四章介绍了如何为 RX7i 的内置以太网接口做以太网配置（关于 RX7i 和 RX3i 的以太网通讯和配置，详见 PAC 系统的 TCP/IP 以太网通讯手册，GFK2224）

第五章介绍 CPU 操作

六到十章和附录 A 介绍编程特性:.

- 第六章　应用程序原理
- 第七章 程序数据
- 第八章 指令设定参考
- 第九章 服务请求函数
- 第十章 PID 函数
- 第十一章 结构文本

第十二章介绍以太网和串口通讯

第十三章介绍串行 **I/O,SNP** 和 **RTU** 协议

第十四章介绍中断处理

附录 **A** 介绍指令计时

附录 **B** 介绍用户内存分配

附录 **C** 介绍了如何将 **90** 系列 **PLC** 程序转化为 **PAC** 程序，并且概括了两种控制系统操作上的不同点

新特点：

注意：*本手册描述了下列新特点。这些特点并不一定在每个 PAC CPU 上都适用。要确定某一特点是不是适用于特定的 CPU 模式和固件版本，请参考 CPU 附带的重要产品信息（IPI）文件。*

Modbus TCP/IP 服务器

PAC 系统以太网接口最多支持 16 个 Modbus TCP/IP 服务器同时连接到 ETM 模块(最多 16 个 TCP 连接，每个 Modbus 连接上面有一个)。Modbus TCP/IP 服务器允许网络上的 Modbus TCP/IP 用户与 PAC 控制器进行初始化数据传输。SRTP, FTP 和 Web 上的 Modbus TCP/IP 连接是分别管理的。

在 Modbus 和控制器件进行地址空间映射是在 CPU 硬件配置中完成的，在第三章描述。关于 Modbus TCP/IP 服务器操作，参考 PAC 系统 *TCP/IP 通讯*，GFK-2224。

可中断的闪存读/写

闪存读/写操作过程中闪存或者 RAM 的内容是作为独立的文件来拷贝的。编程软件中显示拷贝操作的进程，使你可以随时中断闪存读/写操作，而不必等到整个传输过程进行完。更多信息参考第五章“闪存操作”

I/O 中断触发的附加存储器类型

全局变量%AQ 和%I 以及%AI 一样可以用于 I/O 中断来触发程序块

RX3i 控制器

RX3i 控制器支持 RX3i PROFIBUS Slave 模块，IC695PBS301。见 *PACSystems RX3i PROFIBUS* 模块用户手册，GFK-2301。

关于其他的新的 RX3i 模块，见 PACSystems RX3i 安装手册，GFK-2314。

RX7i 控制器

RX7i 控制器支持新的 300 瓦，24 VDC 电源, IC698PSD300.

关于 RX7i 电源，机架和 I/O 模块，参考 *PACSystems RX7i* 安装手册，GFK-2223

PAC 控制系统总览

注意: 某一特征可能不一定适用于所有的 PAC 系统 CPU。又确定某一特征是否适用于某类型的 CPU 和固件版本，请参考 CPU 附带的*重要产品信息(IPI)*文件

PAC 控制器的环境包含性能，工作能力，开放性和可塑性（flexibility）几个方面组成。PAC 系统整合了 GE FANUC 已有系统的先进技术，能够有效的利用您前期在 I/O 和应用开发方面的投资。

PAC 系统的编程软件为 Machine Edition。Machine Edition 是一个通用的开发环境，您可以使用 Machine Edition 对 PAC 系统进行软件编程，硬件配置和故障诊断。经过编程和配置后，PAC CPU 可以作为机器，进程和原料处理系统的实施控制器使用。CPU 通过机架上的背板总线与 I/O 和智能模块通讯。与编程器或 HMI 设备通讯则有三种模式可供选择：1.使用以太网口（RX7i 系统以太网口可能内置）进行通讯 2.使用 GE FANUC 的 SNP-X, Serial I/O 或 RTU slave 协议通过串口 1，串口 2 进行通讯。

PAC CPU 类型

产品类别	产品编号	描述
RX3i	IC695CPU310	300MHz CPU
RX7i	IC698CPE010	300Mhz CPU 内置以太网口
	IC698CPE020	700Mhz CPU 内置以太网口
	IC698CRE020	700MHz 冗余 CPU 内置以太网口

PAC CPU 具有以下共性的特点：

■梯形图和 C 语言编程

■浮点数功能

■可配置的数据和程序存储器

■使用以电池做后备电源的 10 兆字节 RAM 存储器存储用户数据（程序，配置，寄存器数据和符号变量）

■使用 10 兆字节闪存存储用户数据（程序，配置，寄存器数据和符号变量）。闪存为可选配置。

■使用电池保护程序，数据和当前时间（TIME OF DAY CLOCK）时钟

■运行/停止模式转换可配置

■嵌入式的 RS-232 和 RS485 通讯

■最多 512 个子程序块，每个子程序块最大 128KB

■自动为符号变量分配地址，不必在新建变量时，人工为变量指定地址

■通过变量表%W 访问海量存储器区域。海量存储器最大可配置到用户 RAM 的上限。

■和 90 系列 CPU 相比，变量表更大：数字量输入输出（%I 和%Q），为 32K 位。

■模拟量输入输出（%AI 和%AQ），为 32K 字。

■Test Edit 模式使用户能够更容易的测试对正在运行程序的更改效果

■能够单独判断保持型寄存器字中的某一位的状态，使你能够将这一位做为二进制表达方式和功能块输入输出参数。

■系统内固件可更新

RX3i 总览：

RX3i 控制系统硬件包含一个 RX3i 机架和最多七个 90-30 系列扩展机架。CPU 必须放在主机架（RACK 0）上，但可以放在最后一槽之外的任意一个槽上。最后一槽为串行总线发送器保留，IC695LRE001。

RX3i 支持用户定义功能块（只适用于梯形图逻辑）和结构化文本编程。

RX3i 机架使用的双背板总线提供：

■更高的速度，新型高级 I/O 的 PCI

■串行背板，已有的 90-30 系列 I/O 可以快捷转换

The RX3i 通用背板和 90-30 系列扩展/远程机架支持 90-30 系列 Genius 总线控制器和动作控制模块，以及大多数的 90-30 系列/RX3i 前缀为 IC693 和 IC694 的离散和模拟 I/O. RX3i 目录号的前缀为 IC695，其中包括以太网和其他只能安装在通用背板上的通讯模块。所支持的模块的列表见 *PACSystems RX3i 系统手册，*GFK-2314。

通讯特点包括:

■开放的通讯，支持以太网和串口协议。以太网接口（在背板槽上）有两个 RJ-45 口，两个 RJ-45 接口之间通过自动识别开关连接。机架和机架之间不需要转换开关或 HUB。以太网接口支持上传，下装和在线监控功能，提供 32 个 SRTP 通道，并且允许 48 个 SRTP 服务器同时连接。关于以太网接口容量的详细情况参考 PAC 系统 TCP/IP 以太网通讯 GFK-2224

■ RX3i 通过 PROFIBUS 主控器模块支持 PROFIBUS 通讯 。细节详见 *PACSystems RX3i PROFIBUS* 模块用户手册， GFK-2301.

■ 两个串口，一个 RS232 口和一个 RS-485 口

RX7i 总览

RX7i 控制系统硬件包含一个 RX7i 机架和最多七个 90-70 系列扩展机架。CPU 必须放在主机架（RACK 0）的第一槽上。

RX7i 机架使用 VME64 背板。VME64 的带宽达到当前已有的基于 VME 的系统（包括 90-70 系列）的带宽的四倍以上。VME64 支持所有标准 VME 模块，包括 90-70 系列 I/O 和 VMIC 模块。

扩展机架支持 90-70 系列数字量和模拟量 I/O,Genius 总线控制器和高速计数器。CPU 提供嵌入式的自适应的 10/100Mbps 全/半双工以太网接口。

RX7i 支持热备份（HSB）CPU 冗余，从而使一些要求苛刻的程序或进程能够在其中一个出现故障的时候能够继续正常运行。

CPU 冗余系统包含一个激活的控制单元和一个备份控制单元，备份控制单元与激活的控制单元同步并且在必要时可以替代激活的控制单元对进程的控制。每一个单元必须有一个冗余 CPU，IC698CRE020。冗余通讯通道由 IC698RMX016 冗余存储转换模块设置为冗余连接。关于 RX7i 冗余系统的操作，详见 PAC 系统热备份冗余 CPU 用户手册，GFK-2308

通讯特点包括：

■开放的通讯，支持以太网,Genius 和串口协议。

■CPU 上的内置 10/100mb 以太网接口（在背板槽上）有两个 RJ-45 口，两个 RJ-45 接口之间通过自动识别开关连接。机架和机架之间不需要转换开关或 HUB。以太网接口支持上传，下装和在线监控功能，具有基本的利用网络浏览器对控制系统的远程监控功能，最多支持 16 个 WEB 和 FTP 连接。关于以太网接口容量的详细情况参考 PAC 系统 TCP/IP 以太网通讯 GFK-2224

■两个串口，一个 RS232 口和一个 RS-485 口

■一个 RS232 隔离的以太网 站管理器串口

其它系统到PAC 系统的转换

PAC 控制系统提升了已有系统的消费比。支持已有的 90-70 系列模块和扩展机架，从而保护了您的硬件投资。在不改变面板接线的情况下就可以完成升级。

RX3i 支持多数 90-30 系列模块和扩展机架。RX3i 支持的 I/O,通讯，运动（MOTION）和智能模块列表详见 PAC RX3i 安装手册 GFK-2299

RX7i 支持多数 90-70 系列模块，扩展机架和 Genius 网。RX7i 支持的 I/O,通讯和智能模块列表详见 PAC RX7i 安装手册 GFK-2223

允许将 90-30 系列和 90-70 系列的程序转化为 PAC 程序，保护已有的开发成果。

VersaPro 和 LogicMaster 应用软件转化为 Machine Edition 使用户能够平滑转换到 PAC 系统

PAC 系统文献

PAC 系统手册

- PAC 系统 CPU 参考手册，GFK2222
- PAC 系统 TCP/IP 以太网通讯 GFK-2224
- PAC 系统站管理器 GFK-2225
- PAC 系统 C 工具包用户指南 GFK-2259
- RX3i 手册
- PAC 系统 RX3i 硬件与安装手册 GFK-2314
- RX7i 手册
- PAC 系统 RX7i 硬件与安装手册 GFK-2223
- PAC 系统 RX7i VME 模块集成用户指南 GFK-2235
- PAC 系统 RX7i 存储交换模块用户手册 GFK-2300
- PAC 系统 CPU 冗余手册 GFK-2308
- Genius 总线控制器用户手册 GFK-2017
- Proficy Machine Edition 逻辑开发-PLC 起步 GFK-1918
- 模拟量输入，64 通道，16 位 IC697VAL264 模块用户指南 GFK-2056
- 模拟量输出，32 通道，12 位 IC697VAL301 模块用户指南 GFK-2058
- 模拟量输入，带隔离，16 通道 IC697VAL132 模块用户指南 GFK-2060
- 数字量输入，IC697VDD100 模块用户指南 GFK-2062
- 延时输出，64 点 IC697VDR151 模块用户指南 GFK-2063
- 数字量输出，64 点 IC697VDQ120 模块用户指南 GFK-2066
- 8 通道 RTD/Strain 规格 IC697VRD008 模块用户指南 GFK-2098
- 90 系列手册
- C 可编程协处理器模块和支持软件 GFK-0255
- 90 系列串行通讯驱动器用户手册 GFK-0582
- 90 系列 C 语言编程工具用户手册 GFK-0646
- 标准化安装要求 GFK-1179
- 90 系列 PLC 站管理器的 TCP/IP 以太网通讯手册 GFK-1186
- 90-70 系列可编程控制器安装手册 GFK-0262
- 90-70 系列 CPU 指令设定参考手册 GFK-0265
- 90-30 系列 CPU 指令设定参考手册 GFK-0467
- 90-70 系列 Genius I/O 系统用户手册 GEK-90486-1

90-70 系列 Genius I/O 模拟量和数字量模块用户手册 GEK-90486-2

Chapter 2 CPU 特点及说明

本章详细介绍了 PAC 系统 CPU 的硬件特点并对这些硬件的特点进行了说明

PAC 系统CPU 的共同特点

闪存中的固件存储

CPU 用闪存来储存操作系统固件。所以在固件升级时不用拆开模块，也不用更换 EPROM。只需用兼容 PC 机通过串口连接 PLC，运行包含升级包的软件即可.

操作，保护和模块状态

CPU 操作可以使用三位的运行/停止（RUN/STOP）开关控制，也可以用编程器和编程软件远程控制。编程和配置数据可以用软件密码锁定。（RX7i CPU 用一个指示灯指示以太网的状态）。详见每个类型 CPU 的指示器（”Indicators”）

注意：RESET 按钮用于支持以后的某些特征，对当前版本的 CPU 操作没有影响

以太网全局数据（EGD）

以太网全局数据

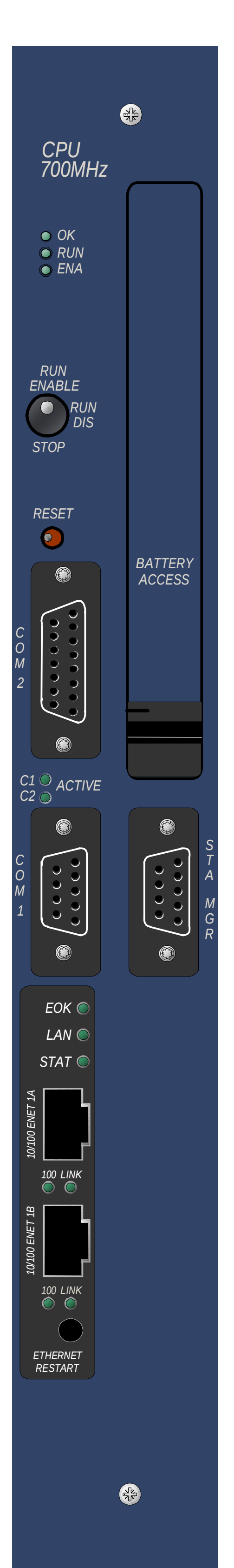
每一个 PAC CPU 通过自己所有的以太网接口最多可以同时支持 255 个以太网全局数据（EGD）页，以太网全局数据（EGD）页必须在编程软件中配置并且存储到 CPU 中，以太网全局数据（EGD）配置也可以从 CPU 上传到编程软件中。Produced 页和 Consumed 页都可以配置。PAC CPU 只支持一部分 Consumed 太网全局数据（EGD）页的使用。太网全局数据（EGD）页只能发送或接受内网的数据。

PAC CPU 发送太网全局数据（EGD）页的最小周期为 2 毫秒，判断发送超时的时间精度同样为 2 毫秒。使用时可以将发送周期设定为 0，这表示每一个输出扫描时发送一次，这种“越快越好”的方式的最小发送周期为 2 毫秒。

在太网全局数据（EGD）的配置过程中，PAC 系统的机架/槽将自动识别以太网接口。

RX7i 特点及说明

- **IC698CPE010:** 300MHz CPU 处理器
- **IC698CPE020:** 700MHz CPU 处理器
- **IC698CRE020:** 700MHz CPU 带冗余处理器



指示器:

五个 CPU 指示灯指示 CPU 在不同功能时的操作状态

以太网接口指示器由七个发光二极管组成（LEDs）。七个指示灯都为绿颜色指示灯，由以太网接口控制。

■模块正常（EOK）

■已连接到局域网上（LAN）

■状态（STAT）

■2 个激活的指示灯（LINK）

■2 个速度指示灯（100）

EOK,LAN 和 STAT 组合起来指示以太网的状态。

每一个以太网接口由两个指示灯指示状态，这两个指示灯分别为 Link 指示灯和 100 指示灯。Link 指示灯指示网络连接状态和是否激活。物理连接正常时，Link 指示灯点亮；检测到数据流时，Link 指示灯闪烁。注意：指示灯点亮并不一定表示以太网口有数据流，在以太网的转换口上有数据流时，指示灯同样闪烁。100 指示灯指示网络数据传输速度（10 或 100Mb/秒）。如果网络连接速度为 100Mbps，100 指示灯点亮。

指示灯操作在下表描述：

CPU 指示灯操作

指示灯状态 ●On ✱Blinking ○Off			CPU 操作状态
●	**OK**	On	CPU 通过上电自诊断程序，并且功能正常
○	**OK**	Off	CPU 有问题，EN 和 RUN 指示灯可能以错误代码模式闪烁，技术支持可据此查找问题。应将这种情况和任何错误代码报告给技术支持
✱	**OK**, **EN**, **RUN**	有节奏的闪烁	CPU 在启动模式，等待串口的固件更新信号
●	**RUN**	On	CPU 在运行模式
○	**RUN**	Off	CPU 在停止模式
●	**EN**	On	输出扫描使能
○	**EN**	Off	输出扫描失效
✱	**Port 1** **Port 2**	闪烁 闪烁	端口信号激活

*在完成初始化步骤之后

以太网指示灯操作：

指示灯状态 ●On ✱Blinking ○Off			以太网操作状态
✱ ○ ○	EOK LAN STAT	以错误代码闪烁 Off Off	硬件故障
✱ ○ ○	EOK LAN STAT	快速闪烁 Off Off	正在进行自诊断
✱ ○ ○	EOK LAN STAT	慢速闪烁 Off Off	等待来自 CPU 的以太网配置
✱ ● ✱ ○ ✱	EOK LAN STAT (*EOK 和 STAT 指示灯有节奏的闪烁)	慢速闪烁* On/Traffic/Off 慢速闪烁*	等待接收 IP 地址
● ✱ ○	EOK LAN STAT	On On/Traffic/Off On/Off	可操作的
✱ ✱ ✱	EOK LAN STAT (*所有指示灯有节奏的闪烁)	慢速闪烁* 慢速闪烁* 慢速闪烁*	加载软件

EOK 指示灯操作

EOK 指示灯指示以太网接口是否能够执行正常的操作。发生硬件故障或者不可恢复的运行故障时，指示灯以两位数字代码闪烁，向用户指示对应的错误代码。指示灯闪烁时首先显示最重要的那个错误代码，之后停顿一下，显示相对不太重要的错误代码，再停顿较长的一段时间后，重复以上显示过程。

以太网硬件故障的 EOK 指示灯闪烁代码

闪烁代码	描述
0x12	未定义中断或意外中断
0x13	上电自诊断过程中的定时器故障
0x14	上电自诊断过程中的 DMA 故障
0x21	上电自诊断过程中的 RAM 故障
0x22	上电自诊断过程中的堆栈错误
0x23	上电自诊断过程中的共享存储器借口错误
0x24	上电或工厂测试过程中的固件循环冗余校验（CRC）错误
0x25	运行异常
0x31	未定义指令或零除溢出
0x32	软件中断
0x33	放弃预取指令
0x34	放弃数据
0x35	运行时的意外中断请求（IRQ）
0x36	运行时的意外的快速中断请求（FIQ）
0x37	保留的例外情况或者过零分支

* 正常操作时循环冗余校验（CRC）错误或软件错误将导致以太网重启

LAN 指示灯操作：

LAN 指示灯指示以太网的访问状态。正常操作或者等待接收 IP 地址时，LAN 指示灯闪烁表示正在访问网络。指示灯保持常亮则表示虽然以太网口不能正常访问网络，但网络可用。LAN 指示灯指示的网络可用意味着以太网物理接口可靠并且有一个或两个以太网口连接到网络上。

STAT 指示灯操作

STAT 指示灯指示在 Normal 操作模式下的以太网接口环境。若 LAN 指示灯灭，则有事件进入到异常日志中，用户可以通过站管理器接口看到异常事件。没有异常事件时，STAT 指示灯常亮。

在其他状态时，STAT 指示灯关断或者闪烁以定义模块操作状态。

以太网口指示灯操作（100Mb 和 Link/Activity）

每个以太网口各有两个绿颜色指示灯，100 和 LINK。100 指示灯指示网络数据传输速度（10 或 100Mb/秒）。如果网络连接速度为 100Mbps，100 指示灯点亮。

Link 指示灯指示网络连接状态和是否激活。物理连接正常时，Link 指示灯点亮；检测到数据流时，Link 指示灯闪烁。注意：指示灯点亮并不一定表示以太网口有数据流，在以太网的转换口上有数据流时，指示灯同样闪烁。

以太网重启按钮

用这个按钮可以手动重启以太网固件，从而不必重启整个系统来实现以太网的重启。

重启时的指示灯操作

在任何状态下手动按按钮重启以太网时，EOK,LAN 和 STAT 指示灯首先都有节奏的闪烁以测试指示灯本身的好坏。这三个指示灯点亮 1/2 秒的时间，并且在固件重启时熄灭。手动重启以太网固件不会影响以太网接口指示灯。

只有在手动重启时才会检测指示灯，用站管理器重启时不会对这三个指示灯进行检测。

串口

CPU 前面板上有三个独立的串口。串口 1，2 既可供外部设备做串行连接使用，也可以做固件升级使用。第三个串口做以太网 站管理器口使用。所有的串口是隔离的。关于串口针脚和串行通讯的详细情况，详见第十二章。

以太网口

在内置的以太网口上有两个 RJ-45 以太网接口，任何一个都可以访问其他以太网设备，也可以同时利用这两个接口访问其他以太网设备。每一根网口自动检测通讯速率（10 或 100Mb/秒），双工模式（半双工或全双工）以及电缆连接方式（直接电缆连接或交叉电缆连接）。关于以太网口针脚的详细情况，详见第十二章。关于以太网通讯的详细情况，参考以下手册：
PAC 系统 TCP/IP 以太网通讯 用户指南 GFK-2224

PAC 系统站管理器 手册 GFK-2225

警告

以太网接口的两个 RJ-45 接口不能直接或间接连接到同一设备上。同一以太网上的 HUB 连接或转换连接必须形成树状，否则可能会产生重复的数据包

错误校验与纠正

冗余 CPU IC698CRE020 具有错误校验与纠正（ECC）功能。使用错误校验与纠正（ECC）功能会轻微影响系统性能。因为在上电过程中系统要对额外的 8 位数据进行初始化，所以在上电过程中这个影响相对显著。若想将非冗余 CPU IC698CPE020 的固件更新，使之支持冗余功能，则须根据升级包中的指令将错误校验与纠正（ECC）跳线跳到使能状态。PAC 系统的其他 CPU 不具有这个功能。
关于错误校验与纠正（ECC）的详细内容，参考 PAC 系统热备份 CPU 冗余用户指南，GFK-2308

关于 RX7i CPU 的说明

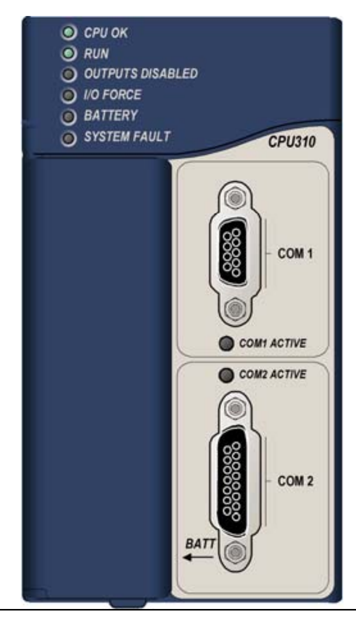
关于环境的说明，参考 RX7i 安装手册附录 A 中的 “RX7i 通用说明”，GFK-2223

电池寿命：存储保持能力	关于不同情况下的电池寿命，参考2-错误！未定义书签。页
程序存储	多达 10M 字节的以电池做后备电源的 RAM 10M 字节的稳定的用户闪存

电量需求 CPE010 CPE020, CRE020	+5 VDC: 名义电流 3.2A +12 VDC:名义电流 0.042A -12 VDC:名义电流 0.008A +5 VDC: 名义电流 4.5 A +12 VDC: 名义电流 0.042 A -12 VDC: 名义电流 0.008 A
使用温度	CPE010：不带风扇 0~50°C（32°F~122°F） 带风扇 0~60°C（32°F~140°F） CPE020,CRE020：必须带风扇，0~60°C（32°F~140°F）
浮点	有
典型的布尔数执行速度 CPE010 CPE020	每 1000 个布尔触点/线圈 0.195ms 每 1000 个布尔触点/线圈 0.195ms
当前时间时钟精确度 通电时间时钟精（内置计时器）确度	每天最大误差为 9 秒 最大误差 0.01%
内置通讯	RS232,RS485,以太网接口
支持的串口协议	Modbus RTU slave,SNP,串行 I/O 要确定是否可用于某一特定的固件版本，参考随 CPU 提供的重要产品信息手册
以太网口	内置自动识别的 10/100Mbps 全/半双工以太网接口
VME 兼容性	系统支持 VME64 标准 ANSI/VITA1
程序块	最多 512 个程序块。每个子程序块最大 512K
存储器（详细的存储范围列表，参考第七章）	%I 和%Q：32K 位离散量 %AI 和%AQ：最多可配置为 32K 字 %W：最大可配置到 RAM 的上限 符号变量：最多配置 10M 字节
错误校验与纠正	只适用于 CRE020

以太网接口说明	
基于 Web 的数据监视	最多 16 个 Web 服务器和 FTP 连接（组合的）
以太网通讯速率	10/100Mbps
物理接口	10 Base T RJ45
WinLoader 支持	YES
EGD 配置页数	255
时间同步	SNTP
EGD Cosumption 是否可选择	YES
从 PLC 向编程器装载数据	YES
UDP 上的远程站管理器	YES
本地站管理器（RS-232）	专用的 RS-232 口
可配置的高级用户参数	YES

RX3i 特点及说明



- IC695CPU310: 300 MHz CPU 处理器

串口

CPU 前面板上有 2 个独立的串口。串口 1，2 既可供外部设备做串行连接使用，也可以做固件升级使用。关于串口针脚和串行通讯的详细情况，详见第十二章。

指示器:

8 个 CPU 指示灯指示 CPU 在不同功能时的操作状态

指示灯操作在下表描述：

CPU 指示灯操作

指示灯状态 On　闪烁　Off		*CPU 操作状态*
On	**CPU OK** On	CPU 通过上电自诊断程序，并且功能正常
Off	**CPU OK** Off	CPU 有问题，允许输出指示灯和 RUN 指示灯可能以错误代码模式闪烁，技术支持可据此查找问题。应将这种情况和任何错误代码报告给技术支持
闪烁	**CPU OK**, 允许输出, **RUN** 有节奏的闪烁	CPU 在启动模式，等待串口的固件更新信号
On	**RUN** On	CPU 在运行模式
Off	**RUN** Off	CPU 在停止模式
On	允许输出 On	输出扫描使能
Off	允许输出 Off	输出扫描失效
On	**I/O** 强制 On	位变量被覆盖
闪烁	电池 闪烁	电池电量过低
On	电池 On	电池失效或没安装电池
On	系统故障 On	CPU 发生致命故障，在停止/故障状态
闪烁	**COM1** 闪烁 **COM2** 闪烁	端口信号可用

*在完成初始化步骤之后

说明

关于环境的说明，见 *PACSystems RX3i 系统手册*附录 A，GFK-2314.

IC695CPU310	
电池：存储保持能力	见 2-错误！未定义书签。页，不同条件下的电池寿命.
程序存储	多达 10M 字节的以电池做后备电源的 RAM 10M 字节的稳定的用户闪存
电源要求	+3.3 VDC: 名义电流 1.25 A +5 VDC: 名义电流 1.0 A
使用温度	0~60℃（32℉~140℉）
浮点	有
典型的布尔数执行速度	每 1000 个布尔触点/线圈 0.195ms
当前时间时钟精确度	每天最大误差为 2 秒
逝去时间(内部定时器)精度	最大 0.01%
内置通讯	RS232,RS485
支持的串口协议	Modbus RTU slave,SNP,串行 I/O
背板	支持 2 种背板总线：PX3i PCI 和 90-30 串行
PCI 兼容性	按 PCI 2.2 标准设计
程序块	最多 512 个程序块。每个子程序块最大 512K
存储器（详细的存储范围列表，参考第七章）	%I 和%Q：32K 位离散量 %AI 和%AQ：最多可配置为 32K 字 %W：最大可配置到 RAM 的上限 符号变量：最多配置 10M 字节

估计电池寿命

为避免 RAM 存储器内容丢失，应定期更换 CPU 的锂电池。下表所列举的电池寿命数据将有助于制定电池更换周期表：

IC698ACC701 电池寿命

控制器	*平均温度*	*有电源供电时的名义寿命:*	
		总是有外部电源供电	*无外部电源供电*
IC698CPE010 IC698CPE020 IC698CRE020	20°C (68°F)	5 年	40 天
IC695CPU310	20°C (68°F)	5 年	40 天

IC698ACC701 电源在平均的温度 20°C (68°F)下存放时间为 5 年

注意: 外部辅助电源模块 IC693ACC302 可以为任何的 PACSystems CPU 提供更长时间的备份电源。细节请参考辅助电源模块数据页 GFK-2124

Chapter 3

CPU 配置

用 Machine Edition Logic Developer 软件配置 PAC CPU 和 I/O 系统。

CPU 在上电时检查实际的模块和机架配置，并在运行过程中定期检查。实际的配置必须和程序中的配置相同。二者之间的配置差别作为配置故障报告给 CPU 报警处理器。关于配置功能，参考 **Machine Edition Logic Developer-PLC** 入门手册（**GFK-1918**）和在线帮助。

注意：可以通过安装不同的固件并且更改跳线设置的方法便捷的将 IC698CPE020 转化为 CRE020。CRE020 的固件更新包里有详细说明。

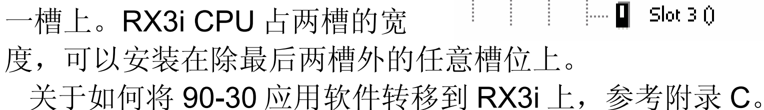
CPU 配置

如图，使用 **Developer-PLC** 编程软件配置 CPU 的步骤如下：

RX7i
Data Watch Lists
Ethernet Global Data
Hardware Configuration
Rack 0 (IC698CHS017)
Slot 0 (IC698PSA100)
Slot 1 (IC698CPE020)
Ethernet
Slot 2 (Used With Slot 1)
Slot 3 ()

1. 依次点开浏览器的 Project/PAC Target/hardware configuration/main rack(rack0)条目
2. 右键点击 CPU Slot，选择 Configure。软件弹出参数编辑窗口，其中显示 CPU 参数。

   注意：RX7i CPU 只能安装在第一槽上。RX3i CPU 占两槽的宽



度，可以安装在除最后两槽外的任意槽位上。

关于如何将 90-30 应用软件转移到 RX3i 上，参考附录 C。

3. 编辑参数值 点击想要配置的键，选择正确的配置参数输入区。关于这些配置参数输入区，参考 3-2 页的“配置参数”
4. 将配置存储到 PLC 中使这些设定发挥作用。详细信息参考 3-15 页的“存储（下装）配置”

   注意：内置以太网接口在 CPU 槽的子槽上显示。关于如何配置内置以太网接口，参考第四章

配置参数：

设置参数：

这些参数说明了 CPU 的基本操作特性。关于这些参数会对 CPU 的运行产生什麽样的影响，参考第五章。

设置参数（Setting Parameters）	
Passwords （密码）	指明是否使用了密码功能　缺省值：使用（enabled） **注意：** 设为不使用密码时，要想改为使用密码，必须先将 PLC 内存清空
Stop Mode I/O Scanning （停止模式 **I/O** 扫描）	指明 PLC 在停止模式下是否进行 I/O 扫描。缺省值：不扫描（disabled） 注意：90-70 系列 PLC 的这个参数对应 I/O ScanStop 参数
看门狗定时器（毫秒） **(Watchdog Timer(ms))**	（增加值为 10ms 的整数倍）数值必须大于程序扫描周期。看门狗定时器用于检测“无法完成扫描”故障。CPU 在每次扫描开始时刻重启看门狗定时器。扫描过程中看门狗定时器进行时间累加。看门狗定时器用于检测应用程序的非正常执行情况，应用程序的非正常执行会导致 PLC 的单次扫描无法在看门狗定时器设定的时间内完成。 正确范围: 10 ~ 1000, 增加值最小为 10. 缺省值: 200. **注意:** 关于如何在 CPU 冗余系统中设定看门狗定时器，参考 PAC 系统热备份 CPU 冗余用户指南, GFK-2308.
Logic/Configuration Power-up Source **(**上电时的逻辑**/**配置来源**)**	指明上电后从哪里调用逻辑/配置数据（或将哪里的数据下装/拷贝到 RAM） 选项：总使用 RAM, 总使用 Flash，有条件使用 Flash(Always RAM,Always Flash,Conditioal Flash) 缺省值：总使用 RAM （Always RAM）
Data Power-up Source 上电时的数据来源	指明上电后从哪里调用变量数据（或将哪里的数据下装/拷贝到 RAM） 选项：总使用 RAM, 总使用 Flash，有条件使用 Flash(Always RAM,Always Flash,Conditioal Flash) 缺省值：总使用 RAM （Always RAM）
运行**/**停止模式转换 （**Run/Stop Switch**）	使用或不使用运行/停止模式转换 选项： 使用（Enabled）:使用户能够利用 PLC 上的物理开关将 PLC 从运行状态切换到停止状态再切回运行状态，以便清除非致命错误 不使用（Disabled）：不使用 PLC 上的物理开关 缺省值：使用（enabled） 注意：如果两个窗口都没有配置为 RTU Slave 或 SNP 协议，就必须有其他停止 CPU 的方法，或者通过其他设备（比如以太网模块）控制 CPU，否则无法将运行/停止模式开关（Run/Stop Switch）设为不使用（disabled）模式。如果 CPU 被设定为停止（Stop）模式，那末他使用的协议将从串行 I/O 协议转换为停止模式协议（缺省为 RTU Slave 协议）。关于停止模式设定，参考 3-10 页的“1 口，2 口参数（Port1,Port2 Parameters）”
存储器保护转换 **(Memory Protection Switch)**	是不是使用存储器保护转换(Memory Protection Switch)设置跟运行/停止模式转换（Run/Stop Switch）有关系： 使用（Enabled）:不允许覆盖程序和配置数据，不允许强制和覆盖离散数据 不使用（Disabled）:不使用存储器保护功能 缺省值: Disabled.

设置参数（Setting Parameters）	
上电模式 （**Power-up Mode**）	选择的 CPU 模式在上电后立刻生效 选项：Last,Stop,Run 缺省：Last(最后一次掉电时，本模式生效)
Modbus 地址空间映射类型	指明用于在 PAC 控制器和 Modbus TCP/IP 之间传输数据时的地址空间映射类型 选项： **Disabled:** “Disabled” 设定倾向于用在包含老的以太网固件，不支持 Modbus TCP 的系统。 **标准 Modbus 地址:**使以太网固件使用标准映射，显示在 Modbus TCP 地址映射键下 缺省: Disabled 关于 PAC 系统 Modbus/TCP 服务器的执行，参考 PAC 系统 *TCP/IP 通讯*, GFK-2224

Modbus TCP 地址映射

只读键显示了 Modbus 地址空间和 CPU 地址空间之间的地址映射分配。PAC 控制器的所有以太网模块和子板基于这种映射使用 Modbus-to-PLC 地址映射。

Modbus 寄存器 **r**	Modbus 协议使用五种指定的变量: 0xxxx 线圈表.映射到 CPU 的 %Q 变量表 1xxxx 离散输入表. 映射到 CPU 的 %I 变量表 3xxxx 输入寄存器表. 映射到 CPU 的 %AI 寄存器变量表. 4xxxx 保持寄存器表. 映射到 CPU 的 %R 变量表. 6xxxx 文件访问表. 映射到 CPU 的 %W 变量表.
起始地址	列举了映射区域的起始地址
结束地址	列举了映射区域的结束地址。对于字存储器类型(%AI, %R 和%W)可用的最高地址在 Memory 键下配置。
PLC 存储器	列举了映射区域的存储器类型.
长度	显示了映射区域的长度

扫描参数

这些参数定义 CPU 扫描执行的特征

扫描参数	
扫描模式	扫描方式定义了各种任务在 CPU 执行过程中的优先级，以及分配给各个任务的时间。参数可以根据扫描方式进行修改。 根据 PLC 扫描方式的不同，控制器通讯窗口(Controller Communication Window)，背板通讯窗口(Backplane Communication Window)和后台窗口(Background Window)以不同的模式运行 选项: ■Normal 模式：PLC 尽可能快的完成扫描，扫描时间依赖于逻辑程序和窗口进程要求。此模式下总的扫描时间等于逻辑执行时间和各个窗口定时器数值的总和。完成自己的所有任务后，窗口关闭。这是一个缺省值。 ■Constant Window 模式：每一个窗口都以 Run-to-Completion 模式模式执行。PLC 按照窗口定时器的设定参数轮流执行三个窗口。总的 PLC 扫描时间等于窗口定时器设定值加上程序执行时间。这个时间会因程序逻辑的执行时间不同而不同。 ■Constant Sweep 模式：总的 PLC 扫描时间固定。一次扫描过程中，某些窗口-----甚至所有的窗口都可能不被执行。本次扫描时间达到扫描定时器的设定值时，窗口停止执行。
逻辑检查字数	每个扫描周期作为输入量给到 checksum 表的用户逻辑字 正确范围：0~32760，最小增量为 8 缺省值：16
控制器通讯窗口模式	（只在扫描方式设定为 Normal 时可用）实现对控制器通讯窗口(Controller Communication Window)的设定 选项： ■Complete:窗口执行完为止，没有时间限制。 ■Limited:时间段方式。控制器通讯窗口(Controller Communication Window)在每个扫描周期内的执行时间由控制器通讯窗口定时器(Controller Communication Window Timer)参数决定 缺省值：Limited 注意：90-70 系列 PLC 对应的参数名称为编程器窗口模式（Programmer Window Mode）
控制器通讯窗口定时器(毫秒)	（只在扫描方式设定为 Normal 时可用。控制器通讯窗口模式(Controller Communication Window Mode)设定为 Complete 时，不允许更改此参数）。每个周期控制器通讯窗口(Controller Communication Window)的最大执行时间应小于看门狗定时器 (Watchdog Timer(ms))设定值 正确的范围和缺省值依赖于控制器通讯窗口模式(Controller Communication Window Mode)： ■Complete:没有时间限制。 ■Limited: 正确范围：0~255ms .缺省值：10 注意：90-70 系列 PLC 对应的参数名称为编程器窗口定时器（Programmer Window Timer）

扫描参数	
背板通讯窗口模式	（只在扫描方式设定为 Normal 时可用）实现对背板通讯窗口(Backplane Communication Window)的设定 选项： ■Complete:窗口执行完为止，没有时间限制。 ■Limited:时间段方式。背板通讯窗口(Backplane Communication Window)在每个扫描周期内的执行时间由背板通讯窗口定时器(Backplane Communication Window Timer)参数决定 缺省值：Complete
背板通讯窗口定时器（毫秒）	（只在扫描方式设定为 Normal 时可用。背板通讯窗口模式(Backplane Communication Window Mode)设定为 Complete 时，不允许更改此参数）。每个周期背板通讯窗口(Backplane Communication Window)的最大执行时间可以大于看门狗定时器 (Watchdog Timer(ms))设定值 正确的范围和缺省值依赖于背板通讯窗口模式(Backplane Communication Window Mode)： ■Complete:没有时间限制。背板通讯窗口定时器(Backplane Communication Window Timer)参数只能读取 ■Limited: 正确范围：0~255ms .缺省值：255（冗余 CPU 10ms）
后台通讯窗口定时器（毫秒）	（只在扫描方式设定为 Normal 时可用）。每个周期后台通讯窗口(Background Window)的最大执行时间应小于看门狗定时器 (Watchdog Timer)设定值 正确范围：0~255 .缺省值：5ms(冗余 CPU)
扫描定时器（毫秒）	（只在扫描方式(Sweep Mode)设定为 Constant Sweep 时可用）每个周期 PLC 最大扫描时间不能大于看门狗定时器 (Watchdog Timer(ms))设定值 一次扫描过程中，某些窗口---甚至所有的窗口都可能不被执行。PLC 本次扫描的时间达到扫描定时器(Sweep Timer(ms))的设定值时，窗口停止执行 正确范围：5~2550 最小的增加值为 5。如果键入的值不是 5 的倍数，将自动转换为邻近的比较大的那个 5 的倍数值 .缺省值：100
窗口定时器（毫秒）	窗口定时器（毫秒）
最后的扫描数	（只适用于具有 1.5 或更高版本固件的 CPU）在运行过程中收到停止指令时，CPU 应该先完成几次扫描，再转到停止模式。（只适用于 Stop 和 Stop Fault，不适用于 Stop Halt） 选项：0，1，2，3，4，5 缺省值： 创建一个新的 PAC 系统对象（Target）时：0 将 90-70 系列 PLC 的对象（Target）转为 PAC 系统对象（Target）时：0 将 90-30 系列 PLC 的对象（Target）转为 PAC 系统对象（Target）时：1

存储器参数

PAC 系统用户存储器包含应用程序，硬件配置(HWC)，寄存器(%R)，海量存储器(%W)，模拟量输入(%AI)，模拟量输出(%AQ)和符号变量

符号变量特征使用户不用在创建变量时人工为其确定存储位置（如同典型高级语言的变量定义）。关于符号变量的使用参考第七章

分配给应用程序和硬件配置的存储容量由实际的程序（包括 C 逻辑数据，%L,%P），硬件配置（包括 EGD 和 AUP）和创建的符号变量决定。其它的用户存储器可根据应用程序配置。例如：一个应用程序可能个头很大，但只使用很少的寄存器变量和模拟量输入/输出变量。同样的，也有逻辑程序很小，但使用很多寄存器变量和模拟量输入/输出变量的情况。附录 B 概括介绍了如何计算用户存储器

计算符号变量所需的存储空间

存储符号变量所需的字节数如下：

((离散的符号变量的位数)/(8 位/字节))+(字类型的符号变量个数)*(2 字节/字)

注意：考虑到强制和其它的位变量特征，上式离散符号变量的位数应为实际个数*4

全部的用户存储器配置计算：

全部的用户可配置的存储器容量（字节数）按下式计算：

全部符号变量所占用字节+全部变量字数*（2 字节/字）+[如果使用点故障功能](存储器内的全部%AI 变量的字数+存储器内的全部%AQ 变量的字数)*(1 字节/字)+ [如果使用点故障功能](存储器内的全部%I 变量的位数+存储器内的全部%Q 变量的位数)/(8 位/字节)

注意：全部变量点只考虑系统内存，没考虑用户存储器。

存储器分配

存储器参数	
点变量	
数字量输入%I，数字量输入%Q, 内部数字量%M；系统变量%S, %SA,%SB 和%SC；临时状态变量%T，Genius 全局变量%G	每个存储类型的存储量上限。只读
字变量	
模拟量输入(%AI)	正确范围：0~32640 缺省值：64
模拟量输出(%AQ)	正确范围：0~32640 缺省值：1024
寄存器变量(%R)	正确范围：0~32640 缺省值：1024
海量存储器变量(%W)	正确范围：0~可用的用户 RAM 的上限。增加值最小为 2048 个字 缺省值：0
符号变量存储器	
离散型符号变量(位)	配置可用的离散型符号变量位数 正确范围：0~83,886,080 改动时，数据改变量应为 32768 的整数倍 缺省值：32768.
非位类型的符号变量(字)	为符号型非离散变量配置 16 位寄存器数 正确范围：0~5,242,880 改动时，数据改变量应为 2048 的整数倍 缺省值：65536
点故障存储器	
点故障变量	如果想在程序逻辑中使用故障（FAULT）结点，就必须使用点故障变量（Point Fault Reference）参数。CPU 要为点故障变量额外分配内存。 在一次操作过程中同时选择向 PLC 中下装硬件配置和逻辑时，CPU 将在下装过程中检查逻辑中是否有故障结点，如果有，CPU 会检查硬件配置中的点故障变量（Point Fault Reference）参数是否为使能（Enabled）状态，如果此参数为失效（Disabled）状态，反馈区会显示错误信息。 向 PLC 中下装逻辑时，CPU 将在下装过程中检查逻辑中是否有故障结点，如果有，CPU 会检查硬件配置中的点故障变量（Point Fault Reference）参数是否为使能（Enabled）状态，如果此参数为失效（Disabled）状态，反馈区会显示错误信息。

故障参数

用户可将每一个故障类型配置为诊断型(diagnostic)故障和致命(fatal)故障
诊断型(diagnostic)故障不会导致 PLC 停止运行。只会设定一个诊断变量并将其记录到故障表上
致命(fatal)故障会使 PLC 转到故障停止模式。致命(fatal)故障也会设定一个诊断变量并将其记录到故障表上

故障参数	
机架丢失	(故障组号 0x01) 发生 BRM 故障，某个机架电源断掉或者找不到配置上的某个机架时，系统变量#LOS_RCK(%SA12) 转为 ON 状态(排除硬件故障并且重新上电后，该变量转为 OFF 状态) 缺省值：诊断型(diagnostic)
I/O 控制器丢失	(故障组号 0x02)总线控制器与 PLC 之间通讯停掉，或者找不到配置上的某个总线控制器时。系统变量#LOS_IOC (%SA13) 转为 ON 状态(更换模块并且将模块所在机架重新上电后，该变量转为 OFF 状态) 缺省值：诊断型(diagnostic)
I/O 模块丢失	(故障组号 0x03) I/O 模块与 PLC 之间通讯停掉，或者找不到配置上的某个 I/O 模块时，系统变量#LOS_IOM (%SA14) 转为 ON 状态(更换模块并且将模块所在机架重新上电后，该变量转为 OFF 状态) 缺省值：诊断型(diagnostic)
选择模块丢失	(故障组号 0x03) 选择 模块与 PLC 之间通讯停掉，或者找不到配置上的某个选择模块时，系统变量#LOS_SIO (%SA15) 转为 ON 状态(更换模块并且将模块所在机架重新上电后，该变量转为 OFF 状态) 缺省值：诊断型(diagnostic)
系统总线错误	(故障组号 0x0C) 背板总线发生错误时，系统变量#SBUS_ER (%SA32) 转为 ON 状态(主机架重新上电后，该变量转为 OFF 状态) 缺省值：致命的
I/O 控制器或 **I/O** 总线故障	(故障组号 0x09)总线控制器报告总线故障，全局存储器故障或者 IOC 硬件故障时，系统变量#IOC_FLT (%SA22) 转为 ON 状态(更换模块并且将模块所在机架重新上电后，该变量转为 OFF 状态) 缺省值：诊断型(diagnostic)
系统配置不等	如果在下装程序或者系统上电时检测到配置不等，系统变量#CFG_MM (%SA9) 转为 ON 状态(在配置相等时将 PLC 重新上电，或者将与硬件相匹配的配置文件下装后，该变量转为 OFF 状态)0x0B 缺省值：致命型(fatal)
Fan Kit Failure	(Fault group 0x17.) When a fault is detected in the Smart Fan kit, system variable #FAN_FLT (%SA7) turns ON. (To turn a fan kit fault OFF, clear the PLC fault table or reset the PLC.) Default: Diagnostic.
可恢复的内存错误	只适用于冗余 CPU. (故障组号 0x26)。确定是某一位的 ECC 错误导致 CPU 停掉还是允许 PLC 继续运行 选项：诊断型(diagnostic)，致命的(fatal) 缺省值：诊断型(diagnostic) 注意：多个位发生 ECC 故障时，CPU 硬件故障组(第 13 组)会记录为致命内存故障(错误代码：169)
CPU 过热	(故障组号 0x18，错误代码：0x001)CPU 运行时的温度超过正常值时，系统变量#OVR_TMP (%SA8) 转为 ON 状态(清除 PLC 故障表或者重启 PLC 后，该变量转为 OFF 状态) 缺省值：诊断型(diagnostic)

故障参数	
PLC 故障表大小 **(PLC Fault Table Size)**	(只读)PLC 故障表最多记录的故障数 此值设为 64
I/O 故障表大小	(只读) I/O 故障表最多记录的故障数 此值设为 64

冗余参数（只适用于冗余 CPU）

这些参数只适用于冗余 CPU(比如 IC698CRE020)。关于如何配置 CPU 冗余参数,参考 PAC 系统冗余热备份 CPU 用户指南 GFK-2308.

转换列表

这些参数只适用于冗余 CPU(比如 IC698CRE020)。关于如何配置 CPU 冗余参数,参考 PAC 系统冗余热备份 CPU 用户指南 GFK-2308

端口 1，端口 2 参数

这些参数配置 CPU 串口的操作特性。端口 1，2 具有相同的配置参数。所选择的协议（端口模式）决定了每个端口的可配置参数

端口参数

端口模式 (Port Mode)

串口上使用的协议决定了端口的显示的参数列表：

选项：

- RTU 从属 模式:为使用 Modbus RTU 从属协议预留。这种模式允许 SNP 主控器（比如 Winloader 和编程软件）连接端口
- 消息模式(Message mode): 端口允许用户逻辑访问。在这种模式下，C 语言块可以通过 C 库函数进行串口 I/O 操作
- 可用(Available): PLC 固件不使用此端口
- SNP 从属 模式:为使用 SNP 从属协议预留。这种模式允许 SNP 主控器（比如 Winloader 和编程软件）连接端口。
- 串口 I/O(Serial I/O): 使用户能使用 COMMREQ 函数实现通用的串行通讯功能

  缺省值: RTU 从属(RTU Slave)

选择停止模式协议，设定对应的停止模式参数为 YES，并且为停止模式选择一个协议。如果停止模式协议不同于端口模式协议，用户可以为停止模式协议设定参数。

如果用户没有为停止模式选择协议，系统采用缺省协议和缺省参数

端口(运行)模式	停止模式
RTU 从属 (RTU Slave)	选项: SNP Slave, RTU Slave 缺省值: RTU Slave.
消息模式 (Message Mode)	选项: SNP Slave, RTU Slave 缺省值: RTU Slave.
可用性(Available)	可用(Available)
SNP 从属 (SNP Slave)	SNP Slave
串行 I/O (Serial I/O)	选项: SNP Slave, RTU Slave 缺省值: RTU Slave.

注意: 如果两个窗口都没有配置为 RTU Slave 或 SNP 协议，就必须有其他停止 CPU 的方法，或者通过其他设备（比如以太网模块）控制 CPU，否则无法将运行/停止（Run/Stop）开关设为失效（disabled）模式。如果 CPU 被设定为停止（Stop）模式，那末他使用的协议将从串行 I/O 协议转换为停止模式协议（缺省为 RTU Slave 协议）。如果 SNP 主控器，比如说编程软件为串行模式，开始通过某个端口通讯，RTU Slave 协议自动转换为 RTU Slave 协议。只要能够将 CPU 停下来，端口协议就会自动转换为允许编程器串行连接的协议。

如果以太网模块可用，用户可以将 Machine Edition 编程软件连接到以太网口，用 Machine Edition 编程软件来控制 CPU。

端口参数	
站地址 **(Station Address)**	(只适用于 RTU Slave 协议) RTU Slave 的 ID 正确范围: 1 ~ 247. 缺省值: 1. **注意:** 应该尽量避免将 PAC 控制系统的其他 Modbus 从属设备的地址设为 1， 因为 CPU 的缺省地址为 1。CPU 在两种情况下使用缺省地址： 1. 无配置的情况下上电，缺省地址为 1。 2. 端口模式参数设为消息模式，停止模式协议为 Modbus 协议时，缺省站地址为 1。 以上两种情况下，如果从属板地址设为 1，CPU 响应该从属板的请求时会产生混乱。 **注意:** 第 1 字节的最低有效位必须为 0。例如站地址为 090019010001 时，9 为第 1 字节。
端口通讯速率 **(Data Rate)**	(除 Available 之外的所有端口模式) 端口通讯速率(位/秒) 选项: 1200 比特, 2400 比特, 4800 比特, 9600 比特, 19.2k 比特, 38.4k 比特, 57.6k 比特, 115.2k 比特. 缺省值: 19.2k 比特
数据位 **(Data Bits)**	(端口模式设为消息模式或串行 I/O 模式时可用)串行通讯的每个字之中有多少位。 SNP 通讯的每个字包含 8 位。 选项: 7, 8. 缺省值: 8.
流控制 **(Flow Control)**	(RTU 从属模式, 消息模式或串行 I/O 模式) 端口使用的流控制类型。 选项: ▪ 对于串行 I/O 模式: 不使用 (None), 硬件 (Hardware), 软件(Software)(XON/XOFF). ▪ 对于其他的端口模式: 不使用 (None), 硬件 (Hardware) 缺省值: 不使用 (None) **注意:** 硬件流控制为 RTS/CTS 交叉式的.
奇偶校验 **(Parity)**	(除 Available 之外的所有端口模式) 奇偶用于串行通讯。如果需要使用调制解调器通讯或者与其他 SNP 主控设备通讯，则需更改此参数 选项: 不使用 (None), 奇校验(Odd), 偶校验(Even). 缺省值: 奇校验(Odd)
停止位 **(Stop bits)**	(端口配置为消息模式(Message Mode), SNP 从属 (SNP Slave)模式或串行 I/O (Serial I/O)时可用)串行通讯的停止位数。SNP 使用一个停止位。 选项: 1, 2. 缺省值: 1.
物理接口 **(Physical Interface)**	(除 Available 之外的所有端口模式)The type of physical interface that this protocol is communicating over. **物理接口的类型即** 选项: ■ 2 线(2-wire):只通过一个通路收发数据。发送时，接收器不工作。 ■ 4 线(4-wire):两个通路分别发送和接收数据，发送通道只在发送数据时工作 ■ 4 线发送(4-wire Transmitter on):两个通路分别发送和接收数据，发送通路连续的发出数据。注意这种方式不适合 SNP 多点通讯，因为多点通讯通路上一个时刻只允许一个设备发送数据。 缺省值: 4 线发送(4-wire Transmitter on)
转换延时时间 **(Turn Around Delay Time (ms))**	(只在端口配置为 SNP 从属 (SNP Slave)模式时可用) 转换延时时间是消息接收到下一次消息发送的最小时间间隔。在 2 线(2-wire)模式下，需要设定这个时间来转换数据传输方向。 正确范围: 0~2550, 最小增加值为 10. 缺省值: 0.

端口参数	
超时(秒) (Timeout (s))	(只在端口配置为 SNP 从属 (SNP Slave)模式时可用)从属设备等待接收主控器消息的最大时间。如果在超时(秒) (Timeout (s))参数设定的时间周期内没有收到消息，从属设备将认定为通讯中断，并且开始等待接收主控器的下一个消息。 正确范围: 0 ~ 60 秒 缺省值: 10.
SNP ID	(只在端口配置为 SNP 从属 (SNP Slave)模式时可用) 端口 ID 用于 SNP 通讯。在 SNP 多点通讯情况下：ID 用于识别将要接受消息的单元。如果通讯是点对点的，这个参数可以空着。要改变 SNP ID，单击数值输入区域，输入新的 ID 即可。SNP ID 最多可以有 7 个字符，并且可以包含字母和数字(A~Z,0~9)以及下划线(_)。
确定停止模式 (Specify Stop Mode)	(除 Available 之外的所有端口模式)确定是使用缺省的停止模式还是自己设定。 选项: **No:** 使用缺省的停止模式 **Yes:**缺省模式参数显示出来，用户可以选择停止模式。如果用户为停止模式选择的协议与运行模式的协议相同，则其他的停止模式参数为只读，并且自动将这些参数值与运行模式对应参数值相同。 缺省值: No.
停止模式 (Stop Mode)	(只在确定停止模式(Specify Stop Mode)设为 Yes 时可用)在串口上执行的停止模式协议。如果用户为停止模式选择的协议与运行模式的协议相同，则其他的停止模式参数为只读，并且自动将这些参数值与运行模式对应参数值相同。 选项: ■ SNP 从属 (SNP Slave):为 SNP 从属协议预留。端口模式只能选为 SNP 从属 (SNP Slave) ■ RTU 从属 (RTU Slave): 为 Modbus RTU 从属 (RTU Slave)协议预留 缺省值: ■ 端口模式设为 SNP 从属 (SNP Slave)时: SNP Slave (唯一正确的选择). ■ 端口模式设为 RTU 从属 (RTU Slave), 消息模式(Message Mode)或串行 I/O (Serial I/O)时: RTU 从属 (RTU Slave).
转换延时时间 (Turn Around Delay Time (ms))	(只在端口配置为 SNP 从属 (SNP Slave)模式时可用) 转换延时时间是消息接收到下一次消息发送的最小时间间隔。在 2 线(2-wire)模式下，需要设定这个时间来转换数据传输方向。 正确范围: 0~2550, 最小增加值为 10. 缺省值: ■ 停止模式不同于端口模式时: 0 毫秒. ■ 停止模式与端口模式相同时: 参数为只读型的，并且与端口模式的转换延时时间参数相同。
超时(秒) (Timeout (s))	(只在端口配置为 SNP 从属 (SNP Slave)模式时可用) 从属设备等待接收主控器消息的最大时间。如果在超时(秒) (Timeout (s))参数设定的时间周期内没有收到消息，从属设备将认定为通讯中断，并且开始等待接收主控器的下一个消息。 正确范围: 0 ~ 60 秒 缺省值: ■ 停止模式不同于端口模式时: 10 秒. ■ 停止模式与端口模式相同时: 参数值为只读型的，并且与端口模式的超时(秒)参数相同。

端口参数	
SNP 标识 (SNP ID)	((只在端口配置为 SNP 从属 (SNP Slave)模式时可用))端口 ID 用于 SNP 通讯。在 SNP 多点通讯情况下：ID 用于识别将要接受消息的单元。如果通讯是点对点的，这个参数可以空着。要改变 SNP ID，单击数值输入区域，输入新的 ID 即可。SNP ID 最多可以有 7 个字符，并且可以包含字母和数字(A~Z,0~9)以及下划线(_)。 缺省值: ■ 停止模式不同于端口模式时: 缺省值为空着不填 ■ 停止模式与端口模式相同时: 参数值为只读型的，并且与端口模式的 SNP ID 相同。
站地址 (Station Address)	(只适用于端口模式设为 RTU 从属(RTU Slave)协议的情况) RTU Slave 的 ID 正确范围: 1 ~ 247. 缺省值: ■ 停止模式不同于端口模式时: 1. ■ 停止模式与端口模式相同时: 参数值为只读型的，并且与端口模式的站地址相同。

扫描设定参数

用户可以为不同的 I/O 扫描设定多个设置，每个扫描有唯一的扫描速率即可。总计 32 个扫描，最多可以设 31 组参数。扫描设定 1 为标准扫描方式，每个周期进行 1 次 I/O 扫描。每一个模块的模块配置中有一个扫描设定参数。扫描设定 1 为缺省设定。

扫描设定参数	
数字 **(Number)**	顺序号 1~32 自动指定给每个扫描设定。扫描设定 1 为标准扫描设定预留。
扫描类型 **(Scan Type)**	确定固定扫描时扫描类型使能 (enabled)还是失效 (disabled) 选项: 失效 (disabled), 固定扫描(Fixed Scan) 缺省值: 失效 (disabled)
扫描数 **(Number of Sweeps)**	(只在扫描类型为固定扫描时可编辑)扫描速率设定，双击输入区，选择一个值。值为 0 时禁止 I/O 扫描。 正确范围: 0 ~ 64. 缺省值: 1.
输出延时 **(Output Delay)**	(只在扫描数(Number of Sweeps)设定值不为 0 时可编辑) The number of sweeps that the output scan is delayed after the input scan has occurred. Double-click on field, then select a value. 输出扫描滞后于输入扫描的扫描数。双击数值设定区，选择一个数值： 正确范围: 0 ~ (扫描数(Number of Sweeps) -1) 缺省值: 0.
描述 **(Description)**	(只在扫描类型为固定扫描时可编辑)扫描设定的简要描述(最多 32 个字符)。

耗电参数

编程软件显示 CPU 在每个电压等级下的耗电量(安培)，CPU 所需电量由电源模块提供。

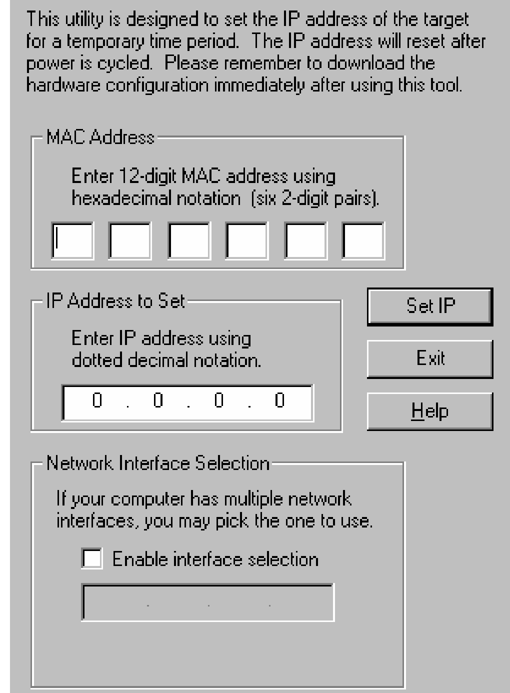
设定临时 IP 地址

第一次使用编程软件与 PAC 系统通讯时，必须首先设定 IP 地址。用户可以使用**设定临时 IP 地址** (Set Temporary IP Address)功能指定 IP 地址，或者通过串口下装带 IP 地址的硬件配置

使用**设定临时 IP 地址** (Set Temporary IP Address)功能，有以下限制：

- 使用**设定临时 IP 地址** (Set Temporary IP Address)功能时，CPU 不能处于运行模式。在 CPU 停止并且不再进行输出扫描之前，不处理通过网络分配的 IP 地址。
- 通过路由器与网络上的 PAC 对象通讯时，**设定临时 IP 地址** (Set Temporary IP Address)功能不起作用。通过转换器或 HUB 与网络上的 PAC 对象通讯时，可以使用**设定临时 IP 地址** (Set Temporary IP Address)功能。
- 当前登陆到计算机上，并且运行**设定临时 IP 地址** (Set Temporary IP Address)功能的用户必须有完整的管理员权限。
- PAC 对象与正在使用**设定临时 IP 地址** (Set Temporary IP Address)功能的计算机必须处于同一个本地子网内。计算机的字网掩码和 IP 地址，以及 PAC 系统的以太网接口的 IP 地址共同确定了这个子网。

注意: 要设定 IP 地址时，必须知道以太网接口的 MAC 地址。



1. 将 PAC 系统连接到以太网上。
2. 浏览器的工程键(Project)下有一个 PAC 系统对象(target)，右键单击单击此对象，选择下线命令，然后选择**设定临时 IP 地址** (Set Temporary IP Address)。将自动弹出**设定临时 IP 地址**对话框
3. **需要在设定临时 IP 地址**(Set Temporary IP Address)对话框内做以下操作：
   - 指定 MAC 地址
   - 在 IP 地址设定框内，输入你想要设定给 PAC 系统的 IP 地址
   - 需要的话，选择**启用网络接口选择校验**(Enable interface selection)对话框，并且标明 PAC 系统所在的网络接口。
4. 以上区域都正确配置之后，单击设定 IP(Set IP)按钮。
5. 对应的 PAC 系统的 IP 地址将被指定为对话框内设定的地址，这个过程最多可能需要 1 分钟的时间。

编程器连接上以后，硬件配置中指定的以太网接口的实际 IP 地址将会下装到控制其中。在进行以下操作前，临时 IP 地址有效：1.以太网接口重新启动或重新上电。2.下装硬件配置或清除硬件配置。

注意

设定临时 IP 地址功能所设定的临时 **IP 地址在重新上电后实效。想要设定永久 IP 地址，必须正确设定对象的 IP 地址并且把硬件配置(HWC)下装(存储)到 PAC 系统中。**

即使对象的以太网接口已经配置了非缺省地址，设定临时 IP 地址功能同样能为其分配临时 IP 地址(包括覆盖编程器先前配置的 IP 地址)

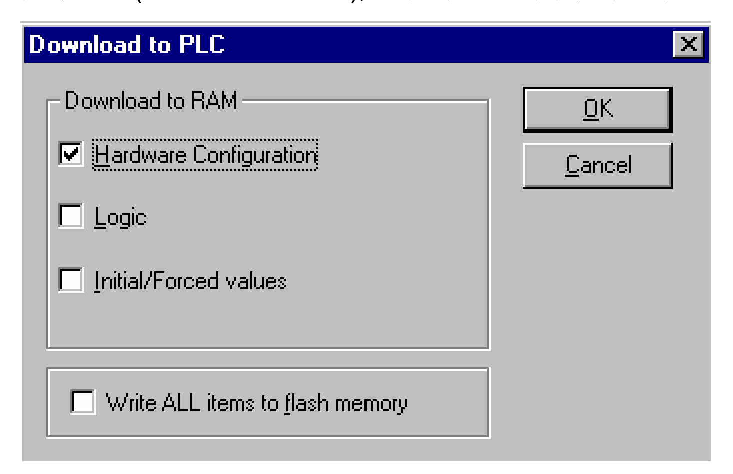
要小心使用这种 IP 地址分配机制

存储(下装)硬件配置

编程软件创建 1 个配置文件，并将这个软件通过串口 1，串口 2 或者以太网接口从编程器传给 CPU。如果使用串口，通讯协议要设为 RTU 从属(RTU Slave)或 SNP 从属 (SNP Slave)

CPU 使用稳定的 RAM 存储器存储配置文件。配置文件存储完成以后，根据新存储的配置参数确定 I/O 扫描的状态(使能(enabled)，失效(disabled))

1. 如果通过以太网接口将硬件配置存储到 PAC 系统中，需要使用初始化 IP 地址功能(见 3-15页)设定以太网接口的 IP 地址。
2. 使 CPU 处于停止模式
3. In Logic Developer-PLC 软件，浏览器的工程(Project)键下，右键单击对象(Target)节点，选择下装到 PLC (Download to PLC), 下装到 PLC 对话框弹出来。



4. 选择想要下装的内容，然后点击 OK

注意: 如果对应 PAC 系统中已有工程，则里面的工程会被覆盖。

Chapter 4

内置以太网接口配置

使用 PAC RX7i 的内置以太网接口前，首先需要使用编程软件配置以太网接口。可以使用软件做以下配置工作：

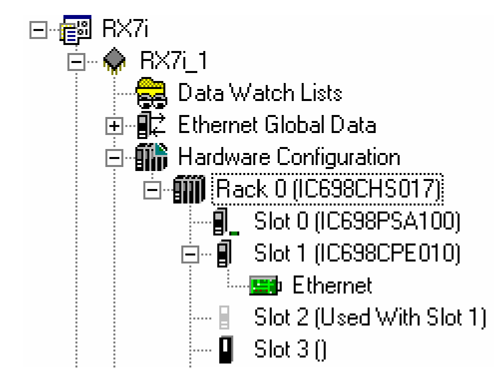
- 定义以太网接口的状态地址
- 给以太网分配 IP 地址，并可选择是否配置子网掩码，网关地址和名称服务器地址。
- 配置站管理器地址(可选)

注意: 只有 RX7i 系列 CPU 提供内置以太网接口。关于如何配置基于机架的 RX7i 和 RX3i 以太网接口模块，参考 PAC 系统的 TCP/IP 以太网通讯，GFK-2224。

配置内置的以太网接口

关于如何使用 Logic Developer-PLC 软件配置机架系统，详见软件的在线帮助。关于 CPU 参数信息，参考第 2 章。

1. 在浏览器的工程键下，点开 PAC 对象，硬件配置和主机架(Rack 0)
2. 点开 CPU 槽(Slot 1)，以太网接口子板显示为“Ethernet”
3. 右键单击子板槽，选择配置(Configure)，参数编辑器窗口显示以太网接口参数。



4. 设定和 RS-232 端口(站管理器)键包含与内置以太网接口功能直接相关的参数。设定键的某些区域必须填写。可以使用 RS-232 键的缺省设置。关于这些输入区的信息，参考 4-2 页的配置参数。建议川口参数设为缺省设置。
5. 保存配置信息并将这些配置信息下装到 CPU 中，以便这些参数发挥作用。

配置参数

以太网参数(设定键)

配置模式	固定为 TCP/IP 模式
适配器名称 **(Adapter Name)**	这个区域设定以太网接口的机架号和槽号，不能更改。对于内置以太网接口，机架号和槽号与 CPU 的相同(0.1.0: 0 机架, 1 槽, 0 号子槽)
为 **IP** 地址使用 **BOOTP (Use BOOTP for Address)**	这个选择指定了是否通过设置 BOOTP 在以太网传输正在工作的 IP 地址。设为假(False : =不使用 BOOTP)时，必须设定 IP 地址(见下面的 IP 地址参数)。设为真(true)时, IP 地址强制设为 0.0.0.0，并且不可编辑。
IP 地址，子网掩码，网关 IP 地址 **(IP Address, Subnet Mask, and Gateway IP Address)**	这些值应由网络管理员分配。如果这些参数不正确，可能导致以太网接口无法进行通讯和(或者)网络操作无法进行。特别重要的是网络上的每个节点的 IP 地址都应该是不同的，不能重复。 然而，如果没有网络管理员或只适用一个简单的没有网关的孤立的网络，你可以如下配置本地 IP 地址： 10.0.0.1　第 1 个 PLC 10.0.0.2　第 2 个 PLC 10.0.0.3　第 3 个 PLC . . . 10.0.0.255　PLC 编程器 TCP or 主机 这种情况下，将子网掩码和网关 IP 地址设为 0.0.0.0。 **注意:**　如果这个孤立的网连接到其他网络，IP 地址不能设为 10.0.0.1 ~ 10.0.0.255，而且子网掩码和网关 IP 地址必须由网络管理员分配。必须分配 IP 地址，以便这 2 个互相连接的网络能够兼容。关于地址的更多信息，*参考 PAC 系统 TCP/IP 以太网通讯手册(GFK-2224)的*"网络管理支持" 也可以参考 4-**错误！未定义书签。** 页的"确定一个 IP 地址是否已在网络上使用" on page.
名称服务器 IP 地址 (**Name Server IP Address**)	域名服务器(DNS)的 IP 地址为 TCP/IP 网络的工作站名和 IP 地址以及 MAC 地址之间建立了关联图。(目前不支持 DNS 特征。建议将参数设为缺省值 0.0.0.0)
Web 服务器的最大连接数 (Max Web Server Connections)	**Web 服务器的最大连接数。**这个数值对应于 web 服务器分配的 TCP 连接的个数。正确范围为 0~16。缺省值为 2。
FTP 服务器的最大连接数 (Max FTP Server Connections)	**FTP 服务器的最大连接数.** 这个数值对应于 FTP 服务器分配的 TCP 连接的个数,而不是 FTP 用户数。FTP 连接建立以后，每个 FTP 用户使用 2 个 TCP 连接。正确范围为 0~16。缺省值为 2
网络时间同步(Network Time Sync)	使网络上的实时时钟同步的方法。现在的选项为：None/DISABLED(不使用网络时间同步功能)，SNTP/ENABLED(与网络上的远端 SNTP 服务器同步)
状态地址 (Status Address)	变量存储器区域接受 LAN 接口状态(LIS)位(16 位)和通道状态位(64 位)。通道状态位紧挨着接口状态(LIS)位存储。状态地址可以分配为%I, %Q, %R, %W, %AI 活 %AQ 存储器。缺省值是下一个可用的%I 地址。 **注意:**　不能将分配给接口状态位和通道状态位的 80 位地址用于其它目的，否则数据会被覆盖。
长度 (Length)	接口状态位和通道状态位的总长度。自动设定为 80 位(状态地址为%I 和%Q 时)或者 5 个字(状态地址为%R, %W %AI, 和 %AQ 时)

冗余(Redundant)IP	(目标 CPU 必须为冗余 CPU，而且至少有一块冗余的存储交换模块-IC698RMX016 做冗余连接)使用这个功能允许以太网接口与对应的双 HWC 对象的冗余硬件配置中的以太网接口共享 1 个冗余的 IP 地址。 **注意:** 可以将系统中的 1 个以太网接口配置为同时使用 Modbus TCP 服务器和冗余 IP 特征。这种情况下，当以太网接口从激活模式转换到备份模式时，除了要停掉冗余 IP 地址，还要断掉所有正在使用冗余 IP 地址的 Modbus TCP 服务器连接。这个特性和 SRTP 服务器一致。结果是任何一次角色转换或者控制权转移都会导致 Modbus TCP 用户程序通讯中断，用户程序需要与冗余 IP 地址重新建立连接来和冗余系统通讯。这种从用户端的操作和正常的丢失与 Modbus TCP 服务器的通讯的情况相同。 缺省值: Disable
冗余(Redundant)IP 地址	(只在冗余 IP 参数设为使能(Enable)时使用)指定给冗余机架中两个冗余以太网接口模块的 IP 地址。(1 个在主 PLC 中，另 1 个在次 PLC 中)虽然冗余 IP 地址由两个以太网接口模块共享，只有激活的以太网接口对这个地址做出响应。 冗余 IP 地址负责你们的网络的人分配。TCP/IP 网络管理员熟悉这些参数并且能够为已经存在的网络分配一些值。这些值应由网络管理员分配。如果这些参数不正确，可能导致以太网接口无法进行通讯并且可能导致网络通讯中断。 正确范围: x.x.x.x, x 大于等于 1，小于等于 255. 缺省值: 0.0.0.0. **注意:** 每个冗余以太网接口有专有的并且是唯一的 IP 地址，用于与其他模块通讯。冗余 IP 地址不能与以太网接口专有地址相同。 **注意:** 关于冗余 CPU 系统的详细信息，参考 PAC 系统 CPU 冗余用户指南，GFK-2308
I/O 扫描设定 (I/O Scan Set)	分配给以太网子板的扫描设定。扫描设定在 CPU 的扫描设定键内定义。正确的范围是 1~32，缺省值为 1。

RS-232 端口(站管理器)参数

以下参数用于 RS-232 站管理器串口

波特率 (Baud Rate)	**端口数据速率(位/秒)。选项为** 1200**,** 2400**,** 4800**,** 9600**,** 19.2k, 38.4k, 57.6k, 115.2k**.**
奇偶校验 Parity	用于端口的奇偶校验类型。选项为不使用，奇校验和偶校验。
流控制(Flow Control)	端口使用的流控制类型。选项为不使用 (None)和硬件(Hardware)
注意:	硬件流控制是 RTS/CTS 交叉式的
停止位	串口通讯的停止位数。选项为 1 和 2

高级用户参数

高级用户参数(AUP)是以太网接口的附加配置参数 。缺省值选为适用大多数用户；只在特殊情况下复杂用户使用时需要修改缺省 AUP 参数。以太网接口的缺省 AUP 值可以通过创建 AUP 文件来更改。，使用编程软件将 AUP 文件输入到以太网接口的硬件配置中，并且将配置存储到 PLC 中。关于如何创建 AUP 文件，参考 PAC 系统以太网接口通讯，GFK-2224.

注意

IEEE 802.3 标准建议不要手动为 1 个端口配置双工模式(由于可能使用高级用户参数)。手动使用 AUP 为端口配置双工模式前，确定你知道连接对象的特性并且知道你所作的选择的结果。根据 IEEE 标准："连接不兼容 DTE/MAU 混合体，比如全双工模式 DTE 与半双工 MAU，或者全双工站(DTE 或者 MAU)连接到中继器或其他的半双工网络"可能导致网络性能严重下降，增加冲突，CRC 错误和未知数据崩溃

注意: 如果强制的对端口的速度和双工模式使用 AUP,转换器将不再进行自动电缆检测。这意味着如果转换端口连接到转换器或者 HUB 端口，就必须使用交叉网线。如果连接到转换器或者 HUB 的上行连接口，或者将转换端口连接到其他以太网设备，你必须使用普通网络电缆。

确定以太网接口是否正常上电

配置完接口以后，按以下步骤操作以确定以太网接口运行正常

1. 关掉电源，3~5 秒后重新上电。这将初始化一系列的自诊断测试。EOK 指示灯将会闪烁以表明上电进程。
2. 正常上电时，以太网指示灯将会如下显示。(关于指示灯的细节，参考第 2 章”CPU 和以太网指示灯”)这时以太网接口在线并且运行正常

指示灯		*以太网接口在线*
EOK	●	On
LAN	● ✲ ○	On, Off, or 闪烁, 根据网络是否激活
STAT	●	On

如果再上电过程中检测到问题，以太网接口不会直接转到操作状态。关于自诊断，参考 *PAC 系统 TCP/IP 以太网通讯，GFK-2224.*

在网络上 PING TCP/IP 以太网接口

PING 是一个测试网络目标是否可访问的程序。PING 程序向这个目标发送一个 ICMP 请求信息并且等待回复。包括 PAC 系统以太网接口的大部分 TCP/IP 节点支持 PING 命令。

你应该 PING 每一个已安装的以太网接口。以太网接口响应 PING 命令不仅表明接口运行并且配置正常，而且表明已经有合适的 TCP/IP 以太网配置信息存储到端口内。

从 UNIX 主机或 PC 上运行的 TCP/IP 软件 Ping 接口

可从 UNIX 主机或 PC 上运行的 TCP/IP 软件都支持 PING 命令(因为大多数 TCP/IP 通讯软件提供 PING 命令)，也可以从其他的以太网接口运行 PING 命令。使用 PC 或者 UNIX 主机时，可以参考 PING 命令，但是一般来讲，需要远端主机的 IP 地址作为 PING 命令的参数。例如，命令应写成如下格式：

```
ping  10.0.0.1
```

确定网络上是否已经使用某一 IP 地址

注意: 这种方法不能保证这个 IP 地址只被网络上的一个设备使用。如果某个网络设备暂时离线，并且与这个 IP 地址相同，这种方法无法检测出来。

IP 地址绝对不能重复。确定其他设备是否使用相同的 IP 地址的方法如下：

1. 断掉本接口与 LAN 的连接。
2. Ping 这个刚断掉的接口的 IP 地址。如果接到应答，表明这个 IP 地址已经被其他节点使用。为避免 IP 地址重复，必须分配唯一的 IP 地址。

Chapter 5 CPU 操作

本章描述 CPU 的操作模式，以及在这些操作模式下 CPU 所担负的任务。将讨论如下题目：

- CPU 扫描
- 程序编制模式
- 窗口模式
- 运行/停止操作
- 闪存操作
- 时钟和计时器
- 系统安全
- I/O 系统
- 上电次序，掉电次序

CPU 扫描

在收到编程器，其他设备，或 CPU 上的运行/停止转换开关发出的停止命令前，CPU 内的应用程序重复执行。除了执行应用程序，CPU 还保存输入设备的输入数据，讲输出数据传给输出设备，做内部清理工作，完成通讯任务以及做自检。这个序列称为**扫描**。

CPU 扫描有以下三种模式：

Normal 扫描	这种模式下，每次扫描时间可以不同。逻辑窗口在每一次扫描过程中完全执行。通讯窗口和后台窗口可设为受限制(Limited)或运行-完成(Run-to-Completion)模式
Constant 扫描	这种模式下，相邻的两次扫描的开始时间间隔由用户设定。逻辑窗口在每一次扫描过程中完全执行。如果还没到扫描周期时间，CPU 轮流执行通讯窗口和后台窗口，一直执行到设定的扫描时间。
Constant Window	这种模式下，每次扫描时间可以不同。逻辑窗口在每一次扫描过程中完全执行。CPU 轮流执行通讯窗口和后台窗口，执行时间由用户确定。

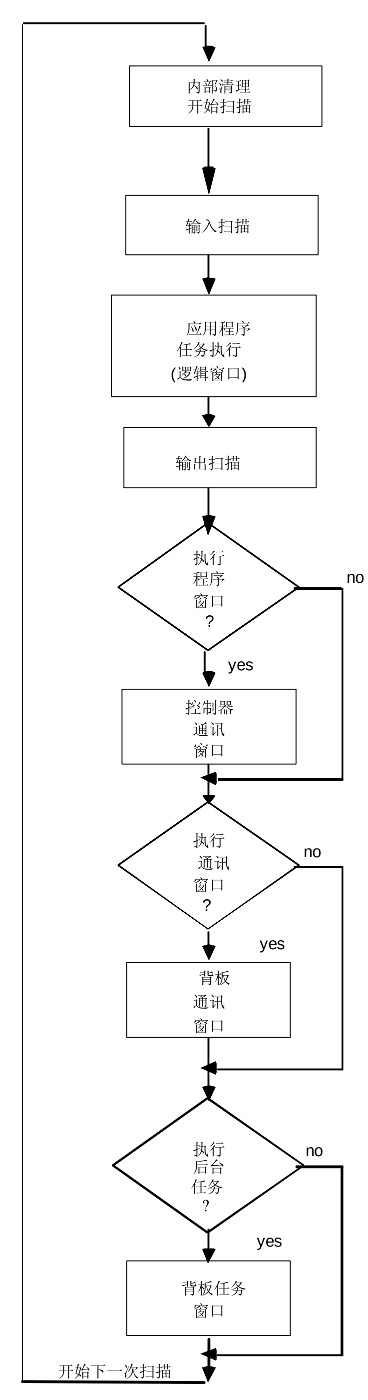
注意:　以上概括叙述了不同扫描模式的一些特征。其他见 5-6 页"CPU 扫描模式"

CPU 以以下四种模式中的一种运行：(详见 5-10 页，运行/停止操作)

- 运行/输出使能
- 运行/不输出
- 停止/IO 扫描
- 停止/无 IO

CPU 扫描的各个部分

典型的 CPU 扫描由下表所示的七个阶段组成：



CPU 扫描的各个部分

典型 CPU 扫描的 7 个主要阶段

阶段	*活动*
内部清理	内部清理部分完成扫描开始前的准备工作。包括更新%S 位，确定定时器的最新值，确定扫描模式(停止或者运行)以及检测扩展机架。 循环检测以确定扩展机架电源是否正常。一旦检测到扩展机架，则扩展机架配置及所有模块数据传到控制器通讯窗口中。
输入扫描	输入扫描过程中，CPU 从 Genius 总线控制器和输入模块读取输入数据。如果数据是从 EGD 页得到的，CPU 会将页内数据从以太网接口拷贝到适当的存储位置。细节见 PAC 系统 TCP/IP 以太网通讯，GFK-2224. **注意:** 如果 I/O 挂起功能激活，那末当前扫描周期中不进行输入扫描。
应用程序执行(逻辑窗口)	CPU 执行程序逻辑时，总是从第 1 条指令开始，执行到最后一条指令终止。执行完最后一条指令后产生新的输出数据。 关于如何控制程序的执行，参考第 6 章。 中断驱动逻辑可以在扫描的任何 1 个阶段执行。详见第 6 章 指令执行时间列表参考附件 A.
输出扫描	CPU 将输出数据写到总线控制器或输出模块。用户程序检查投入使用。 输出扫描过程中，CPU 向 Genius 总线控制器和输出模块写入输出数据。到达 EGD 发送页时间节点时，CPU 从存储器上的对应位置向以太网接口拷贝输出页数据。所有输出数据发出之后，输出扫描完成。 如果 CPU 处于运行模式并且配置为执行后台检测，则后台检测在输出扫描进程的最后时间段内执行。检测字数的缺省值为 16。如果每次扫描的检测字数设为 0，则这个过程跳过。后台检测帮助确认运行模式下的 CPU 内的程序的完整性。 如果 I/O 挂起功能激活，那末当前扫描周期中不进行输出扫描。
控制器通讯窗口	板子上的以太网和串行端口服务。重新配置这部分扫描时的扩展机架和独立模块。 CPU 总是执行这个窗口。执行的窗口条目如下： ■ 重新配置扩展机架和独立模块。控制器窗口内优先级高的重新配置。在分配给这个窗口的时间内，需要的话重新配置模块。重新配置模块需要几个扫描周期。 ■ 通讯活动包括内置以太网接口和 2 个 CPU 串行端口 是否执行控制器通讯窗口和执行控制器通讯窗口的时间可以使用编程软件进行配置。也可以在用户程序中使用**服务请求功能 3** 进行动态配置。窗口时间可以设定为 0~255 毫秒，缺省值为 10 毫秒。 注意如果控制器通讯窗口时间设为 0，由两种方法打开这个窗口：没有电池的情况下断电再上电，或者转到停止模式。
背板通讯窗口	通过此窗口与智能设备进行通讯。基于机架的以太网接口模块在背板通讯窗口通讯。在本阶段扫描过程中，CPU 与 Genius 总线控制器和 TCP/IP 以太网模块等智能模块进行通讯 在这个窗口，CPU 在执行队列中的请求之前，首先完成前面未完成的请求。分配给这个窗口的时间用完以后，进程停止。 背板通讯窗口缺省为完成(运行-完成)模式。这意味着所有智能模块中当前未完成的请求在每个扫描过程中都要处理。这个窗口也可以以限制(Limited)模式运行，这种情况下需设定每个扫描周期内分配给本窗口的最大时间。 模式和时间可以配置并且存储到 CPU 当中，也可以在用户程序中使用**服务请求功能 4** 进行动态配置。通讯窗口时间可以设定为 0~255 毫秒，缺省值为 255 毫秒。扫描时间比较紧张的时候跳过此功能。

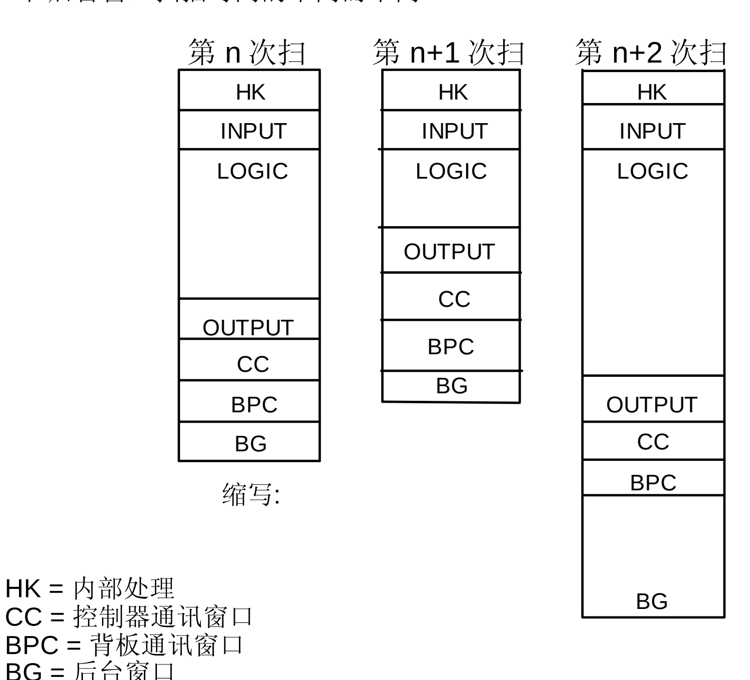
阶段	*活动*
后台窗口	本窗口进行 CPU 自检 本窗口进行 CPU 自检。自检中包括对 CPU 操作系统软件的检测。 后台窗口时间缺省值为 0 毫秒。也可以设定其他的值并存储到 CPU 当中或者通过编程软件在线更改。 后台窗口的 执行和时间也可以在用户程序中使用**服务请求功能 5** 进行动态配置。扫描时间比较紧张的时候跳过后台功能。

扫描模式

Normal 扫描模式

这种模式下，每次扫描时间可以不同。逻辑窗口在每一次扫描过程中完全执行。通讯窗口和后台窗口可设为受限制(Limited)或运行-完成(Run-to-Completion)模式。Normal 扫描模式是控制系统应用程序中最常用的扫描模式。

下表描述了 Normal 扫描模式的 3 个连续的扫描周期。注意整个扫描时间可能因逻辑窗口，通讯窗口和后台窗口扫描时间的不同而不同



Normal 扫描模式的典型扫描

Constant 扫描模式

这种模式下，相邻的两次扫描的开始时间间隔由用户设定。逻辑窗口在每一次扫描过程中完全执行。如果还没到扫描周期时间，CPU 轮流执行控制器通讯窗口，背板通讯窗口和后台窗口，一直执行到设定的扫描时间。到了 CPU 总的 Constant 扫描时间后, 通讯窗口和后台窗口停止执行。

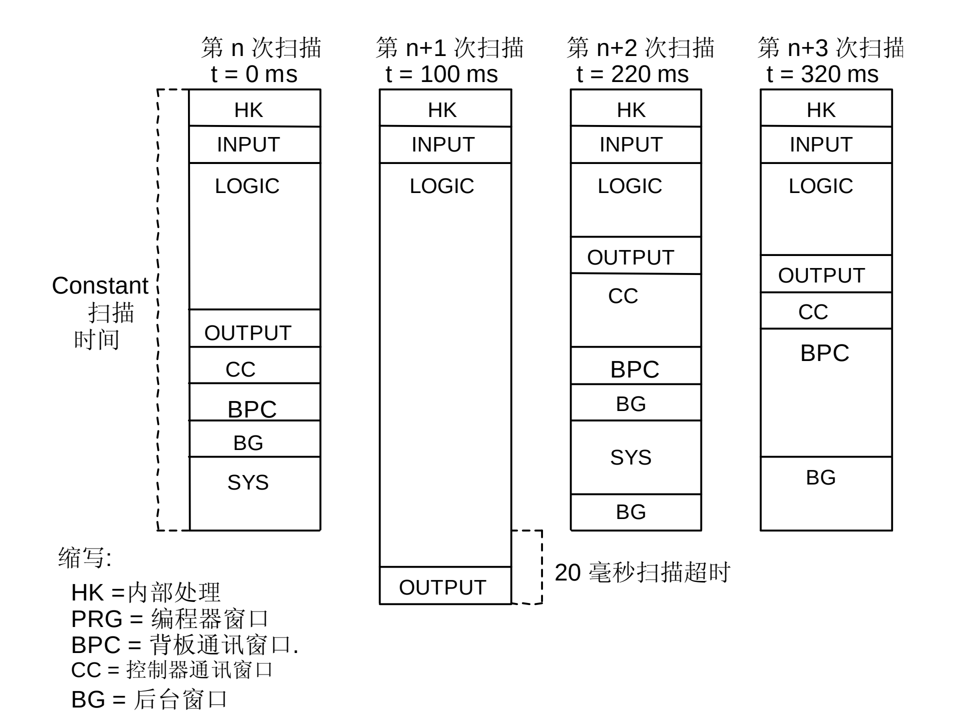
使用 Constant 扫描模式的其中一个原因是确保 I/O 数据定期更新。

Constant 扫描定时器设定范围为 5~2550 毫秒。Constant 扫描周期定时器值和 Constant 扫描模式可以通过编程软件设定或者使用服务请求功能 1 来设定。Constant 扫描定时器没有缺省值；定时器值必须提前设定或者在 Constant 扫描模式设为 Enable 的同时设定。

以太网全局数据可配置为发送和接收，并且会导致扫描时间延长 1 毫秒。配置 CPU Constant 扫描周期或设定 CPU 看门狗时间时，需考虑太网全局数据页的影响。

如果某个周期的扫描时间超过扫描周期设定值，CPU 就会发出扫描超时报警，并在下一个周期开始时设定状态变量 OV_SWP (%SA0002)。扫描超时故障不会影响下一个周期的扫描时间。

下表描述了 Constant 扫描模式的 4 个连续的扫描周期，扫描周期为 100 毫秒。注意整个扫描时间是固定的。但是逻辑窗口扫描时间长于正常值可能导致扫描超时。



Constant 扫描模式的典型扫描

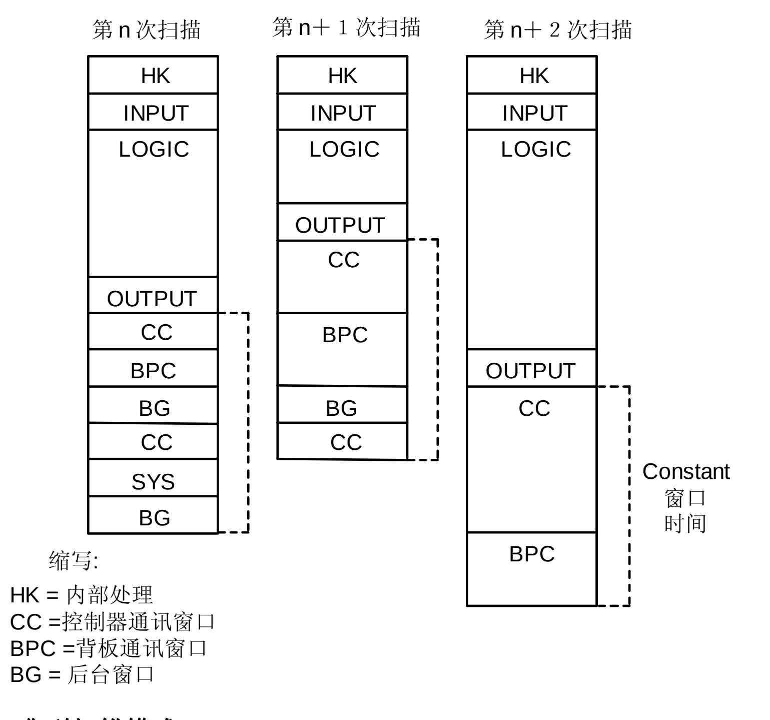
Constant 窗口模式

Constant 窗口模式下，每次扫描时间可以不同。逻辑窗口在每一次扫描过程中完全执行。CPU 轮流执行控制器通讯窗口，背板通讯窗口和后台窗口，执行时间为设定的扫描时间。总计的 CPU 扫描时间等于内部处理时间，输入扫描，逻辑窗口和输出扫描阶段的执行时间以及 Constant 窗口定时器设定时间总和。整个扫描时间可能因逻辑窗口的执行时间的不同而不同。

对于那些要求输入扫描和输出扫描之间有固定的时间间隔或需要在接收到输出数据之后对输入数据作一些设定的应用程序，Constant 窗口模式是一个理想的选择。

Constant 窗口定时器设定值范围为 0~255(毫秒)。Constant 窗口定时器数值可以使用编程软件设定，也可以通过用户程序使用服务请求功能 3、4、5 来设定

下表描述了 Constant 窗口扫描模式的 3 个连续的扫描周期。注意整个扫描时间可能因逻辑窗口扫描时间的不同而不同，但是为通讯窗口和后台窗口指定的时间是常数。因为窗口时间为常数，所以一个周期内某些窗口可能跳过，暂缓或者执行多次。



Constant 窗口典型扫描模式

程序编制模式

CPU 支持 1 种程序编制模式，顺序模式。逻辑窗口中的顺序模式的程序在 1 个扫描周期内完整的执行 1 次

窗口模式

上一部分描述了典型 CPU 扫描的各个阶段。控制器通讯，背板通讯和后台窗口可以基于 CPU 扫描模式以不同的模式运行.(CPU 扫描模式在 5-6 页详细描述)。可以使用下面三种模式：

运行－完成 (Run-to-Completion) 在运行－完成模式下，窗口启动时发出的所有请求都会响应。这个窗口所有挂起的请求完成以后，CPU 转到扫描的下一个阶段。(不适用于后台窗口，因为后台窗口不处理请求)

固定 (Constant) 窗口模式下，控制器通讯窗口，背板通讯窗口和后台通讯窗口执行时间的总和是固定的。这个时间期满时，即使请求正在执行，窗口也将关闭，而且通讯在下一个周期重新开始。如果本窗口没有挂起的请求，则 CPU 空载并等待其他请求。任何一个窗口选择了 constant 模式，其他的窗口都将自动转为 constant 模式

受限 (Limited) 在受限模式下，窗口运行的最大时间是固定的。这个时间期满时，即使请求正在执行，窗口也将关闭而且通讯在下一个周期重新开始。如果本窗口没有挂起的请求，则 CPU 进入扫描的下一个阶段。

通讯窗口的数据一致性

窗口模式为固定 (Constant)和受限(Limited)时，控制器和背板通讯窗口可能 CPU 扫描模式中较早的执行完成。外部设备(比如 CIMPLICITY HMI)传输数据块时，如果在完成请求前关闭通讯窗口，则可能损害数据块的一致性。这个请求会在下一次扫描时完成；然而，可能有部分数据来自这个扫描，另一部分数据来自下一次扫描。CPU 在 Normal 扫描模式并且通讯窗口在运行－完成模式时，不存在数据一致性问题。

注意: 与停止状态的 CPU 通讯的外部设备读取的是 CPU 停止前最后一次扫描的状态。这可能会误导使用 HMI 系统的操作工，使他们不能正确分辨 CPU 的运行／停止状态。进程表会正确反映正常的进程。

同时也要注意的是，在 CPU 从停止状态转到运行状态之前，非保持型的输出量不会清 0 。

运行／停止操作

PAC 系统 CPU 支持 4 种运行／停止模式。你可以通过以下途径改变这些模式：运行／停止转换开关，编程软件配置，LD 功能块和系统的 C 程序调用。可以通过授权等级、运行／停止转换开关位置、密码等手段限制操作模式的转换。

模式	操作
运行／输出使能 (Run/Outputs Enabled)	运行用户程序并且连续不断的扫描输入并且更新自身的输出，包括 Genius 网和以太网输出。控制器和背板通讯窗口在受限模式、运行－完成模式或者固定模式下运行。
运行／输出失效 (Run/Outputs Disabled)	运行用户程序并且连续不断的扫描输入，但是并不更新自身的输出，包括 Genius 网和以太网输出。自身的输出保存在这种模式的缺省状态下。控制器和背板通讯窗口以受限模式，运行－停止模式或者固定模式运行。
停止／IO 扫描使能 (Stop/IO Scan Enabled)	CPU 不运行用户程序，但是进行输入输出扫描。控制器和背板通讯窗口以运行－停止模式运行。后台窗口限定为 10 毫秒。
停止／IO 扫描失效 (Stop/IO Scan Disabled)	CPU 既不运行用户程序，也不进行输入输出扫描。控制器和背板通讯窗口以运行－停止模式运行。后台窗口限定为 10 毫秒。 **注意：** 在停止模式下，冗余 CPU 的 I/O 扫描总为失效(disabled)状态。

注意： 不能在运行模式下增加%P 和%L 变量表的大小，除非首次使用这两种变量或存储的程序块是一个新的程序块。

CPU 停止模式

CPU 在停止模式时有两种操作模式：

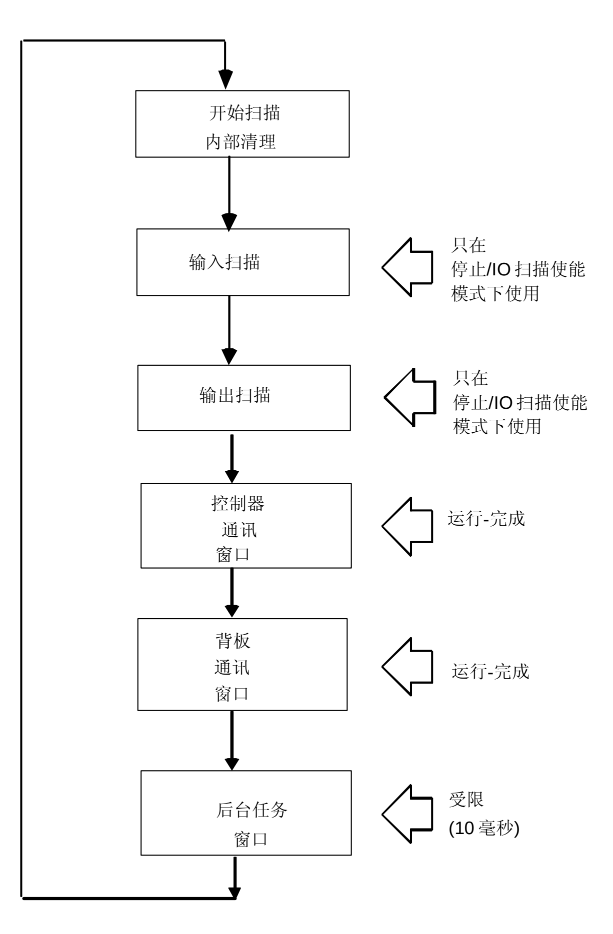
- I/O 扫描使能 – 每个周期都进行输入输出扫描
- I/O 扫描失效 – 跳过输入输出扫描

CPU 在停止模式时不执行应用程序。用户可以自行配置是否在停止模式进行 I/O 扫描。停止模式下，仍然与编程器和智能模块通讯。停止模式下，仍然校验总线接收模块和进行机架重配置。

I 在这两种停止模式中，控制器通讯和背板通讯窗口以运行-停止模式运行，后台窗口以受限模式运行，限定值为 10 毫秒。

最后的扫描数可以在硬件配置中设定。在收到运行到停止的转换信号或者故障停止时，要完成一定次数的扫描才转换到停止状态。缺省值为 0。

服务请求 13(SVCREQ13)用于设定完成多少次扫描后 CPU 转换为停止状态。所有的 I/O 转换到配置的缺省状态，CPU 故障表会记录一条自诊断信息。



停止 I/O 失效与停止 I/O 使能模式下的 CPU 扫描

停止模式到运行模式的转换

CPU 从停止模式到运行模式的转换过程中执行以下操作：

- 确认扫描模式和程序编制模式选择
- 确认程序使用的变量和实际配置的变量范围
- 重新初始化外部程序块和独立 C 程序所使用的数据空间
- 清空非保持型的存储区域

运行/停止模式转换操作

运行/停止模式转换开关有 3 个位置：

转换开关位置	*CPU 和扫描模式*	*存储器保护*
运行 使能(Run Enable)	CPU 运行 I/O 扫描使能	用户程序存储器只读.
运行 失效(Run Dis)	CPU 运行，I/O 扫描失效。	用户程序存储器只读
停止(Stop)	禁止 CPU 转入运行状态	可以向用户程序存储器些数据

可以在编程软件的硬件配置中进行配置，使运行/模式转换开关失效。转换开关的存储器保护功能可以在硬件配置中单独设置，使其失去作用。存储器保护功能缺省值为失效(Disabled)

只要转换开关当前位置所确定的操作模式已经配置了，就可以通过逻辑读取转换开关位置(Switch_Pos)函数读取转换开关的当前位置。详见第八章。

闪存操作

CPU 在以电池为后备电源的 RAM 中存储应用程序和硬件配置。对于 PAC 系统来说，你也可以选择将逻辑，硬件配置，变量数据存入闪存本地闪存。PAC 系统 CPU 提供了足够的闪存空间来存储所有的用户可空间(10MB)里的数据，并且能够满足更高的要求。变量表不计入用户空间。关于哪些内容计入用户空间，请参考附录 B。

CPU 上电时,会读取逻辑和配置，变量表数据。缺省情况是从 RAM 读取。然而逻辑/配置，变量表每一类数据都可以配置为总从闪存读取或有条件从闪存读取。要在编程软件中配置这些参数，选择 CPU 硬件配置中的设定键。

如果选择有条件的将闪存作为的数据来源，只在检测到内存崩溃或者没有电池时从闪存读取数据(上电时)。建议使用此方式，因为闪存是稳定的而且不需要电池作为后备电源来保持数据。

如果选择有条件的将闪存作为上电时的数据来源，在检测到内存崩溃或者程序没保存(没有电池)情况下重新上电，系统会将闪存中的逻辑/配置和/或变量表数据存入 RAM 中。如果逻辑/配置，变量存储器配置为此种模式，但是程序已经有很好的保存，则不进行闪存操作。

如果将逻辑/配置和/或变量表配置为上电时总从闪存读取数据，那麽不管用户存储器是什麽状态，上电时总是将闪存内的逻辑/配置和/或变量表存储到 RAM 中。

注意: 如果任何一个部分(逻辑/配置或变量表)从闪存中读取，那麽 OEM 模式和密码也从闪存中读取。

除了上电时为 CPU 保存逻辑，配置和数据，还可以使用编程软件对闪存进行以下操作：

- 向闪存中拷贝一份当前配置，应用程序和变量表(不会被覆盖)的副本
- 从闪存读取先前存储的配置和应用程序和/或变量表值，并写入 RAM
- 校验闪存和 RAM 中是否存储同样的数据
- 清空闪存内容

闪存读写操作将闪存内容作为一个独立的文件拷贝给 RAM。编程软件显示拷贝操作的进程，你可以随时中止拷贝，而不用一直等到完成传输过程。因为闪存拷贝被认为是成功完成的，所以全部的用户存储器映像必须。如果在向闪存写入数据的过程中取消操作，重新上电或做其它的干涉，CPU 将会清空闪存。如果从闪存读取数据的过程被中断，RAM 将被清空。

上电时的逻辑/配置数据来源和 CPU 的操作模式

闪存和用户存储器可以保存不同的逻辑/配置上电参数。下表概括了这些设定如何在上电后确定逻辑/配置数据来源。CPU 模式受下表所述的上电模式，运行/停止转换和停止模式 I/O 扫描参数，运行/停止转换开关位置，以及断电模式的影响

上电前		*上电后*	
闪存作为上电过程中的逻辑/配置数据来源	***RAM 作为上电过程中的逻辑/配置数据来源***	***逻辑/配置数据来源***	***CPU 模式***
总是使用闪存	存储器内容没保存 (没有电池或内存崩溃)	闪存	见 5-**错误！未定义书签。**页，存储没保存/上电数据来源为闪存时的 CPU 模式
总是使用闪存	RAM 中没有配置，存储器内容已保存	闪存	见 5-**错误！未定义书签。**页，存储保存
总是使用闪存	总是使用闪存	闪存	
总是使用闪存	有条件使用闪存	闪存	
总是使用闪存	总是使用 RAM	闪存	
有条件使用闪存	存储器内容没保存 (没有电池或内存崩溃)	闪存	存储没保存/上电数据来源为闪存时的 CPU 模式
有条件使用闪存	RAM 中没有配置，存储器内容已保存	使用缺省的逻辑/配置	停止/IO 失效 (Stop Disabled)
有条件使用闪存	总是使用闪存	RAM	见 5-**错误！未定义书签。**页，存储保存时的 CPU 模式
有条件使用闪存	有条件使用闪存	RAM	
有条件使用闪存	总是使用 RAM	RAM	
总是使用 RAM	存储器内容没保存 (没有电池或内存崩溃)	使用缺省的逻辑/配置	停止/IO 失效 (Stop Disabled)
总是使用 RAM	RAM 中没有配置，存储器内容已保存	使用缺省的逻辑/配置	停止/IO 失效 (Stop Disabled)
总是使用 RAM	总是使用闪存	闪存	见 5-**错误！未定义书签。**页，存储保存时的 CPU 模式
总是使用 RAM	有条件使用闪存	RAM	
总是使用 RAM	总是使用 RAM	RAM	
闪存中没有配置	存储器内容没保存 (没有电池或内存崩溃)	使用缺省的逻辑/配置	停止/IO 失效 (Stop Disabled)
闪存中没有配置	RAM 中没有配置，存储器内容已保存	使用缺省的逻辑/配置	停止/IO 失效 (Stop Disabled)
闪存中没有配置	总是使用闪存	RAM	见 5-**错误！未定义书签。**页，存储保存时的 CPU 模式
闪存中没有配置	有条件使用闪存	RAM	
闪存中没有配置	总是使用 RAM	RAM	

存储没保存/上电数据来源为闪存时的 CPU 模式

配置参数		*运行/停止转换开关位置*	***CPU 模式***
上电模式	*运行/停止转换*		
Run	Enabled	Stop	Stop Disabled
Run	Enabled	Run Disabled	Run Disabled
Run	Enabled	Run Enabled	Run Enabled
Run	Disabled	N/A	Run Enabled
Stop	N/A	N/A	Stop Disabled
Last	Enabled	Stop	Stop Disabled
Last	Enabled	Run Disabled	Run Disabled
Last	Enabled	Run Enabled	Run Disabled
Last	Disabled	N/A	Run Disabled

存储保存时的 CPU 模式

配置参数			*运行/停止转换开关位置*	*掉电模式*	***CPU 模式***
上电模式	*运行/停止模式*	*停止模式* ***I/O*** *扫描*			
Run	Enabled	Enabled	Stop	N/A	Stop Enabled
Run	Enabled	Disabled	Stop	N/A	Stop Disabled
Run	Enabled	N/A	Run Disabled	N/A	Run Disabled
Run	Enabled	N/A	Run Enabled	N/A	Run Enabled
Run	Disabled	N/A	N/A	N/A	Run Enabled
Stop	N/A	Enabled	N/A	N/A	Stop Enabled
Stop	N/A	Disabled	N/A	N/A	Stop Disabled
Last	Enabled	Enabled	Stop	Stop Disabled	Stop Disabled
Last	Enabled	Enabled	Stop	Stop Enabled	Stop Enabled
Last	Enabled	Enabled	Stop	Run Disabled	Stop Enabled
Last	Enabled	Enabled	Stop	Run Enabled	Stop Enabled
Last	Enabled	Disabled	Stop	N/A	Stop Disabled
Last	Enabled	N/A	Run Disabled	Stop Disabled	Stop Disabled
Last	Enabled	Enabled	Run Disabled	Stop Enabled	Stop Enabled
Last	Enabled	Disabled	Run Disabled	Stop Enabled	Stop Disabled
Last	Enabled	N/A	Run Disabled	Run Disabled	Run Disabled
Last	Enabled	N/A	Run Disabled	Run Enabled	Run Disabled
Last	Enabled	N/A	Run Enabled	Stop Disabled	Stop Disabled
Last	Enabled	Enabled	Run Enabled	Stop Enabled	Stop Enabled
Last	Enabled	Disabled	Run Enabled	Stop Enabled	Stop Disabled
Last	Enabled	N/A	Run Enabled	Run Disabled	Run Disabled
Last	Enabled	N/A	Run Enabled	Run Enabled	Run Enabled
Last	Disabled	N/A	N/A	Stop Disabled	Stop Disabled
Last	Disabled	Enabled	N/A	Stop Enabled	Stop Enabled
Last	Disabled	Disabled	N/A	Stop Enabled	Stop Disabled
Last	Disabled	N/A	N/A	Run Disabled	Run Disabled
Last	Disabled	N/A	N/A	Run Enabled	Run Enabled

时钟和定时器

CPU 提供的时钟和计时器包括逝去时间时钟，当前时间时钟和软/硬件看门狗定时器。

关于 CPU 提供的定时器函数和定时结点，详见第八章“定时器和计数器”

逝去时间时钟

逝去时间时钟记录的是 CPU 通电至目前的时间间隔，发生电源故障此数据不会保持，每次上电这个时钟都会回 0 并重新计时。从这个时钟开始计时到时钟计时计时值自动回 0(中间没有电源故障或停/上电情况，精确到秒)的时间间隔大约为 100 年。

因为逝去时间时钟为系统软件操作和定时器功能块提供了时间基准，所以不能在编程器或用户程序中将其重启。然而，用户程序可以使用服务请求 16 和服务请求 50 读取逝去时间时钟，这个命令可以提供更高的时钟分辨率。

当前时间时钟

硬件的当前时间时钟保存了 CPU 的当前时间。当前时间时钟支持以下 7 种功能：

- 年 (2 位)
- 月
- 日
- 小时
- 分钟
- 秒
- 星期几

当前时间时钟以电池为后备电源，发生外部电源故障时不会清零。可以使用编程软件内的服务请求函数 7 读取/设定时间和日期。

当前时间时钟可以自动处理年与年之间，月与月之间的时间转换，并且可以自动补偿闰年月，一直至 2036 年。

看门狗时间

软件看门狗时间

CPU 内的软件看门狗定时器用于检测不能完成扫描错误。在编程软件中设定软件看门狗定时器数值。允许范围为 10~2550 毫秒，缺省值为 200 毫秒。软件看门狗定时器在每次扫描开始时清零并开始计时。

软件看门狗定时器用于检测应用程序中一些非正常操作，这些非正常使 CPU 不能正常的完成扫描。不正常的应用程序情况如下列例子：

- 模块中回归调用过多
- 循环过多(循环次数过多或者每次的执行时间过长)
- 执行死循环

软件看门狗定时器设定值要比程序最大扫描时间大一点以防止意外超时。对于 Constant 扫描模式，设定软件看门狗定时器数值时应考虑扫描超时的情况。

看门狗定时器在中断定时器执行过程中仍然计时。单个扫描内的中断序列可能引起看门狗定时器达到设定值。

超过软件看门狗时间设定值，OK 灯闪烁，CPU 进入停止/严重故障模式。但是某些功能可执行。一个故障写入 CPU 故障表，输出进入缺省状态。CPU 只通过内置以太网口与编程器通讯，不进行其它的通讯或操作。要恢复，必须将装有 CPU 的机架或背板重新上电。

要使当前的扫描时间扩展到软件看门狗时间之外，可以在应用程序中使用服务请求函数 8 重启软件看门狗定时器。然而，看门狗定时器只能在配置软件中更改。

注意服务请求函数 8 不重启 Genius 总线控制器的输出扫描定时器设置

硬件看门狗定时器

备份回路为 CPU 提供额外保护。如果备份回路激活，CPU 立刻进入重启模式。输出进入缺省状态，不进行任何形式的通讯。CPU 将会停掉。详要恢复，必须重新断电上电。

注意: PAC 系统不支持致命故障的重试。

系统安全

PAC 系统支持以下两种系统安全类型：

- 密码/授权级别
- OEM 保护

密码和授权级别

密码是 PAC 系统 CPU 的一个可配置的特性。可以选择是否使用密码。在编程软件中进行密码设置。编程器在线时，密码为编程器的访问设定了级别。下线模式下，密码失效。

缺省状态是没有密码保护。CPU 的每个权力等级可以有一个密码。密码长度为 1~7 ASCII 个字符。只能使用编程软件更改密码。

设定密码之后，通过任何通讯形式访问 CPU 都要正确的输入相应权力级别的密码。密码通过校验后，访问可以进行本个级别和更低级别的访问(例如，正确输入了第 3 级密码以后，可以访问 1，2，3 级的功能)

注意: CPU 的运行模式转换会覆盖密码。即使在编程器上不能进行运行/停止模式的转换，也可以通过 CPU 上的转换开关进行转换。

权力级别

权力级别	*密码*	*访问描述*
4	Yes	写入配置或者逻辑。配置只能在停止模式下写入，逻辑在运行模式和停止模式下都可以写入。设定或删除任意级别的密码。 **注意:**本级别为不定义密码时的缺省连接级别
3	Yes	CPU 停止模式时写入配置和逻辑，包括逐字转换，增加/删除程序逻辑以及覆盖离散 I/O.
2	Yes	**写到任何存储器。不包括覆盖离散 I/O。可以停止或启动 CPU。可以清空 CPU 和 I/O 故障表。**
1	Yes	读取密码之外的任何 CPU 数据。包括读取故障表，执行数据表，校验逻辑/配置，从 CPU 中上载程序和配置等等。 所有数据都不能更改。这个等级不允许编程器将 CPU 转换到运行模式。

编程器要求的保护等级

编程器要提供新的权力等级以及对应权利等级的密码才能转换到该等级。如果编程器发出的密码与 CPU 访问表内存储的密码不等，则 CPU 拒绝改变权力等级并会在故障表上记录出错信息。这种情况下，会保持当前权力级别不变。转换到没有密码的权力级别时，只需输入这个级别号，密码栏空着即可。转换到低级别和转换到高级别操作方法一样。

取消密码

可以选择是否使用密码保护。不想使用密码的话，可以在编程软件中取消密码。

注意: 没密码时想要设定密码，必须先将 CPU 电池取下来，再重新上电以后才可以设置。

密码保护不允许固件更新。想要进行固件更新，先要取消密码保护，在完成固件更新后，再重新设定密码。

OEM 保护

OEM 保护与密码和权力级别相似。然而，OEM 保护提供了更高的安全级别。用 1~7 位字符作为密码确定使用/不使用 OEM 保护特征。OEM 保护特征使能时，禁止对 CPU 程序进行任何读写操作。

提供了 OEM 密匙作为特别的密码来保护 OEM 商的软件投资。OEM 密匙非空时，CPU 可能处于禁止任何读写的状态。这允许 OEM 商为 CPU 编制控制程序，并且设定 OEM 锁定模式，以防止最终用户读取或修改程序。

注意: 护不允许进行闪存固件更新。想要进行固件更新，先要取消 OEM 保护，在完成固件更新后，再重新设定 OEM 保护。

PAC 系统的 I/O 系统

PAC 系统的 I/O 系统为 CPU 和其他设备提供了接口。PAC 系统的 I/O 系统支持：

- I/O 和智能模块
- Ethernet Interface 以太网接口
- Motion 模块(RX3i)
- Genius I/O 系统 (RX7i)。 Genius I/O 总线控制器 (GBC)为 RX7i CPU 和 Genius I/O 总线提供接口。

I/O 配置

模块识别

除了目录号，编程软件还存储硬件配置中的每个模块传给 CPU 的模块 ID。CPU 用模块 ID 来确定如何与这个模块通讯。

硬件配置下装到 CPU 之后(以及后续的上电过程中)，CPU 会将编程器存储进来的 ID 系统中模块 ID 进行比较。如果模块 ID 不匹配，会产生一个系统配置不匹配故障。

因为相近类型的 I/O 模块可能共用 1 个模块 ID，所以尽管下装的配置中包含目录号与槽内插的模块不匹配，也未必产生系统配置不匹配故障。例如，尽管 32 位输出模块 IC693MDL750 和 IC693MDL924 使用不同的输出逻辑，但是这两个模块的 ID 相同，CPU 无法分辨。

I/O 模块的缺省情况

经过配置，某些输入模块可以向用户程序发出中断信号。缺省情况为，输入模板不发出中断信号并且输入过滤器设为慢速。可以在编程软件中修改(是否允许输入模块发出中断信号)，配置存储完成或重新上电后，这些新的设定生效。

90-70 系列 PLC 的离散输出模块缺省值设定为全部关断。配置工具允许用户将输出缺省模式设定为关断或者保持最后时刻状态。当 CPU 从运行/使能转换到运行/失效或者停止模式以及 CPU 发生致命故障时，设定值生效。

上电时，90-30 系列 PLC 的离散输出模块缺省值设定为全部输出关断。这种缺省状态一直保存到 PLC 的第一次扫描。

关于其他模块在上电和停止模式的特性，参考相关模块的手册。

多次 I/O 扫描设定

PAC 系统 CPU 最多可以有 32 个 I/O 扫描设定。一个扫描设定可以为多个模块设定相同的扫描速率。一个模块只能有一个扫描设定。缺省值是所有模块的扫描设定值为 1，即一个扫描周期内进行一次 I/O 扫描.

对于一些应用程序来说，CPU 逻辑不需要每个周期都获取信息。I/O 扫描设定特性使 I/O 扫描更贴近与他们在用户程序中的应用。如果你有很多 I/O 模块，分别设定模块的扫描特性可以显著的减少扫描时间。

在从停止到运行的转换时，将所有模块扫描设定设为不同值的缺点就显现出来了。这种情况下，带程序延时的扫描设定在第一个周期不进行扫描。直到进行输出扫描时，模块输出才使能，也许是许多个周期之后。因此，在执行用户逻辑和一些模块取消输出之间可能要隔一定的时间。在这段时间内，那些模块的输出处于模块的停止模式状态。停止模式特征随模块不同而不同。一些模块将输出置 0，一些模块保持最后一次扫描时的输出状态，另外一些模块将输出设定为配置好的缺省值。模块输出使能时，模块使用最后一次扫描值，这时输出或者为 0,或者保持变量表中对应的输出值。

Genius I/O

Genius 总线控制器(GBC)控制单个 Genius I/O 总线。任何类型 Genius I/O 设备都可以连接到这个总线上

在 I/O 故障表中，机架，槽，总线，模块和 I/O 点数都有一个故障点。总线号 1 指的是单通道 GBC 的总线。

Genius I/O 配置

编程软件可以配置 Genius I/O 模块的子参数。

Genius I/O 模块的部分参数可以使用 Genius I/O 捕捉监控器来设定。这些参数存储在模块自身的 EEPROM 中。串行总线地址(SBA)可以使用 Genius I/O 手持监控器来设定。关于 Genius I/O 模块的类型，配置和设定信息，参考 Genius I/O 系统用户手册，GEK-90486-1 和 GEK-90486-2。

应用程序可以通过 COMMREQ 功能块请求 GBC 改变某个模块的缺省状态。然而，如果模块不在配置保护模式时，他只接受这种改变。如果设定为配置保护模式，只有使用手持监控器更改缺省状态。90-70 系列 PLC, Genius I/O 的 COMMREQ 函数块的格式在 *90-70 系列 Genius 总线控制器用户手册，GFK-2017 和 90-30* 系列 *Genius 总线控制器用户手册，GFK-1034 中描述*

Genius I/O 数据映射

Genius I/O 离散输入/输出以位的形式存储在 CPU 的位缓存中。Genius I/O 模拟量数据存储在 RAM 的%AI 和%AQ 地址中。模拟量数据每个字(16 位)存储一个通道。

模拟量组合模块(输入和输出数据存储器内)只使用实际的输入输出要占用的地址量。比如 115 VAC 的模拟量组合模块 IC660CBA100 有四个输入两个输出。他使用四个字的模拟量输入，两个字的模拟量输出。

数字量组合模块点数用手持监控器(HHM)配置输入，输出和带反馈输出，占用的地址(%I,%Q)为实际输入输出的 2 倍。因此，不管不管模块上的点怎麽配置，8 个 I/O，115 VAC 的数字量组合模块(IC660CBD100)占用 8 个%I 存储器,8 个%Q 存储器。

模拟量组合模块

6 通道模拟量组合模块包含 4 个模拟量输入通道，2 个模拟量输出通道。当轮到此模块使用 Genius I/O 总线时，他在一个控制消息中广播 4 个通道的输入数据。然后，当轮到 GBC 控制时，GBC 以直接控制信息的方式发出两个输出通道的数据给模块。

低等级模拟量模块

热电偶和 RTD 模块等低等级模块和模拟量组合模块不同，他们只有 6 个通道的输入。

Genius 全局数据通讯

PAC RX7i 支持多种控制系统在同一个 Genius I/O 总线上共享数据。这种机制提供了一种自动并且重复传送%G, %I, %Q, %AI, %AQ, %R 和 %W 数据的方式。使用全局数据不用编写特别的程序，因为全局数据已经集成到 I/O 扫描中了。所有具有 Genius I/O 能力的 GE Fanuc 控制器可以向 RX7i 发送全局数据，也都能从 RX7i 接收全局数据。编程软件用于配置 Genius I/O 总线上全局数据的发送和接收。

注意: RX7i CPU 处于停止/IO 扫描失效模式时，不再进行 Genius 全局数据通讯。然而，如果 CPU 处于停止/IO 扫描使能模式时，仍然进行 Genius 全局数据通讯。

I/O 系统自诊断数据收集

PAC I/O 系统的自诊断数据以以下两种方式保持：

- 如果 I/O 模块有相关联的总线控制器，则由总线控制器向 CPU 提供模块的自诊断数据。关于 GBC 模块的细节，见 5-24，“RX7i 如何处理 GBC 故障”
- 对于不通过总线控制接口的 I/O 模块，CPU 的 I/O 扫描子系统基于模块提供的信息产生一个自诊断位。

自诊断位来自 I/O 模块发送给他们的 I/O 控制器(CPU 或总线控制器)的自诊断数据。自诊断位指示相关模块的故障状态。有故障时，故障位置位，故障清除后，故障位清除。

RX7i 不支持非 GE Fanuc 模块的自诊断数据。应用程序必须使用 BUS Read 功能块来访问这些板子的自诊断信息。

离散 I/O 自诊断信息

CPU 保存每个离散 I/O 点的自诊断信息。应用程序 RAM 中的两个存储器块分配给离散自诊断数据，一个给%I 存储器，一个给%Q 存储器。一个自诊断信息为对应一个 I/O 点。这个数据位说明相关 I/O 数据的正确性。每一个离散点有一个故障变量，可以用两个故障结点监视故障变量状态：故障结点(-[F]-)无故障结点 (-[NF]-)。激活故障结点后，CPU 收集故障数据。下表显示故障结点和无故障结点的状态：

条件	***[FAULT]***	***[NOFLT]***
有故障	ON	OFF
无故障	OFF	ON

模拟量 I/O 自诊断信息

CPU 为模拟量模块和 Genius 块的每个通道建立自诊断信息。每个模拟量 I/O 通道的自诊断信息占用一个字节。每个模拟量 I/O 通道使用%AI 和%AQ 占用两个字节，所以自诊断存储器长度为数据存储器的一半。

模拟量自诊断数据包括自诊断数据和过程数据，过程数据有高报警和低报警位。自诊断数据参考-[F]- 和 -[NF]-结点。过程位参考高/低报警结点，-[HA]- 和-[LA]-。模拟量数据的自诊断数据为每个模拟量输入(字)或者模拟量输出(字)数据的自诊断数据长度为一个字节。模拟量故障结点被应用程序引用时，将对自诊断字节中除过程位所有位进行或运算。任何一个自诊断位为 ON,则报警结点闭合。所有自诊断位 OFF,则报警结点打开。

PAC 系统GBC 故障处理

和故障/丢失 GBC 相关的输入数据的缺省值

丢失 GBC,GBC 不匹配或发生其他故障时，CPU 使用 Genius 总线每个设备的输入的缺省设定(如果将输入据设为缺省)。如果这个设备设为保持最后状态，就不管这个数据了。如果将这个设备设为 OFF,输入数据设为 0。如果使用冗余 GBC，输入数据不受影响。

冗余模块的缺省输入和自诊断数据丢失的应用程序

GBC 报告冗余模块丢失时，CPU 在下一个输入扫描更新用缺省数据更新输入数据表和自诊断表。在下一次输出扫描时更新输出自诊断信息。

上电和掉电顺序

上电顺序

系统上电包含以下部分：

- 上电自检测
- CPU 存储器确认
- 系统配置
- 智能模块自检完成
- 智能模块双端口检测
- I/O 系统初始化

上电自检测

系统上电时，系统中许多模块进行上电自检。CPU 模块执行硬件检测和软件确认检测。智能模块进行安装和确认板子上的微处理器，软件检查确认，本地硬件确认并且通知 CPU 自检完成。任何自检失败都会在系统配置周期的某一时刻以队列形势报告给 CPU。

如果出现电池电量过低指示，电池电量过低故障会记录在 CPU 故障表内

CPU 存储器确认

系统上电的下一阶段是 CPU 存储器确认。首先，系统确认电池电量正常并且以电池为后备电源的 RAM 区域仍然正常。校验电池做后备电源的存储应用程序的 RAM，以确定数据是否已保存。 接着，如果有梯形图程序，要检查_MAIN 块。如果没有梯形图，检查最小的独立 C 程序。

如果系统确认应用程序 RAM 已保存，则会检查某一个位缓存区域，以确认该区域数据是否已保存。如着测试通过，位缓存保持上电值(停止模式转为上电模式时，非保持型输出清 0)。如果检测不正确，或者 RAM 没有保存数据，则认为位缓存出错并将其清空。CPU 这时被清空，和一个新的 CPU 模块刚安装上一样。所有的逻辑和配置文件必须重新从编程器下装到 CPU。

系统配置

完成自检以后，CPU 进行系统配置。CPU 首先将位缓存内所有的系统自检测位清 0。这使以前断电时的故障不会做为故障信息重新出现。然后，CPU 依次检测系统内的每个模块是否已完成自检。

CPU 读取每个模块的信息，与存储的(下装的)机架/槽位配置信息进行比较。实际配置和存储的配置信息的任何不同都会记录到故障表中。

智能模块自检完成

因为要测试通讯和接口设备，智能模块自检时间可能会比 CPU 自检时间长。在完成初始化自检以后， 智能模块会将剩余部分自检时间报给 CPU。这段时间内，CPU 提供必要的附加信息给模块，使模块能够完成自检和配置。如果在自己标明的时间内不报告自检完成，CPU 认为模块发生故障，并且在故障表上留下故障入口。所有的自检完成后，CPU 保存上电时的模块自检信息，并将故障信息 (如果有的话)写入故障表。

智能模块双端口检测

智能模块自检完成并形成最终报告以后，CPU 和智能模块会测试双端口的通讯功能。这些测试确定两个模块可以互相发送和接收数据，同时验证通讯协议所需的中断和旗语能力。在双端口测试完成后，通讯信息系统初始化完成。

I/O 系统初始化

输入模块不需要更深层次配置。输出模块则要进入缺省状态。输出模块在上电和故障模式时的缺省值为所有输出处于 off 状态。也可以另外配置。

总线发送模块会被查询有哪些扩展机架上。基于总线发送模块的响应，CPU 将那些机架和他们的相关槽位列入需要配置的槽位表内。

最后，I/O 扫描器进行初始化。I/O 扫描器先与 I/O 控制器上的每个 I/O 总线建立 I/O 连接，保持这个 I/O 控制器的配置数据，再将这些配置数据与已存储的 I/O 配置数据比较，不同之处会记录到 I/O 故障表中。之后，I/O 扫描器向每个 I/O 控制器传送一份需要 I/O 总线上配置的 I/O 模块列表。I/O 控制器初始化完成以后，I/O 扫描器用程序中的设定替换掉出厂设定。

掉电顺序

电源模块检测到外部交流供电中断超过 15 毫秒时，系统掉电。

电源故障时的存储器数据保持

因为应用程序 RAM 以电池为后备电源，下列数据在掉电/上电过程后会保存下来：

- 应用程序
- 故障表和其他自诊断数据
- 程序和模块检测
- 覆盖数据
- %R 寄存器%L 寄存器和%P 寄存器数据
- 模拟量存储器数据(%AI 和 %AQ)
- 离散输入(%I)的状态
- 保持型离散输出(%Q)的状态
- 保持型离散内部变量(%M)的状态

下列类型的数据在重新上电时不保存：

- 离散临时变量(%T)
- 用在非保持型线圈-()- 上的%M 和%Q 变量
- 系统内部的离散位的状态(系统位，故障位，保留位)

Chapter 6 程序组织

本章提供了 PAC CPU 的程序组织信息：

- 应用程序结构
- 程序执行控制
- 中断驱动模块

PAC 系统应用程序的结构

PAC 系统应用程序是程序块结构的。应用程序中包括系统内的 CPU 和模块所需的所有控制逻辑。

应用程序在编程软件中开发并且传输到 CPU 中。程序存储在 CPU 的稳定内存中。

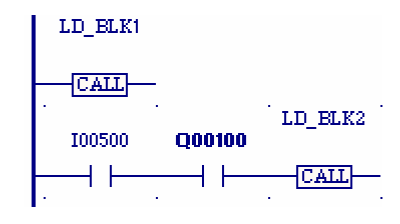
在 CPU 扫描过程中(第 5 章)，CPU 从模块读取输入数据，并且存储在输入存储器中。然后 CPU 使用更新了的输入数据完全执行一次应用程序。应用程序执行后产生新的输出数据并将这些数据写入输出存储器。

应用程序执行完后，CPU 将输出模块写到模块中。

模块结构的程序包含一个_MAIN 块。程序从_MAIN 块开始执行，包括_MAIN 块在内，程序最多可以有 512 个块。

如何调用程序块

在程序逻辑中_MAIN 块和其他块中调用后，程序块执行。在本例中，LD_BLK1 一直被调用。可以设定程序块的调用条件，在输入量%I00500 和输出量 %Q00100 为 ON 时，调用 LD_BLK2。详见第八章的调用函数。



嵌套调用

只要有足够的执行站空间，CPU 就允许进行嵌套调用。如果没有足够的站空间支持程序块调用，会产生一个"堆栈溢出"故障。这种情况下，CPU 不能执行这个程序块。CPU 会将这个模块的所有二进制输出设为 FALSE,并且继续执行程序块调用指令之后的程序。

注意: 由两种方式可以在没有足够堆栈空间时停掉 CPU。最好的方法是增加程序逻辑来测试自诊断位%SA38，以检测用户程序是否发生故障。检测到故障后，可以用 SVC_REQ 13 停掉 CPU。另一个方式是增加逻辑检测模块没有正确执行的信号，然后调用 SVC_REQ 13 停掉 CPU。

除了调用的模块有特别多参数的情况，一般能支持 8 层以上的调用。能进行多少层调用受几个因素的影响，包括程序块数据流量(非布尔型)，程序块调用的特殊功能以及这个块中所定义的参数的类型和数量。如果程序块没有最大限度的使用堆栈资源，就有可能支持多于 8 层的嵌套调用。嵌套调用以_MAIN 块为第 1 层

程序块类型

PAC 系统支持的四种程序块

程序块类型	*本地数据*	*编程语言*	*程序块大小*	*参数*
程序块	有自己的本地数据	LD ST	128 KB	0 输入 1 输出
参数化程序块	从调用者那继承本地数据	LD ST	128 KB	63 输入 64 输出
函数块	有自己的本地数据	LD ST	128 KB	63 输入 64 输出 内部成员变量无限制
外部块	从调用者那继承本地数据	C	用户内存容量极限(10 MB)	63 输入 64 输出

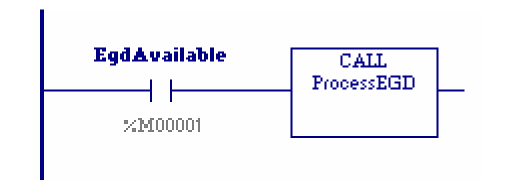
PAC 系统程序块类型自动提供 1 个 OK 输出参数.用来显示模块中的 OK 参数的名称为 Y0。程序块内的逻辑可以读写 Y0 参数。程序块调用时，Y0 参数自动的初始化为 TRUE。除非在程序块内将 Y0 设为 FALSE，否则调用的程序执行完以后会有正电流从调用指令行输出，

对于所有的程序块类型，输入参数的最大数量都比输出参数的最大数量少 1 个。因为 程序块的使能输入不算做一个参数。这个输入被 LD 语言用来确定是否调用这个程序块，但是即使这个块被调用，使能也不作为一个参数进入这个模块。

程序块

任何块都是程序块。创建模块结构程序时。主块自动声明。声明其他任何一个块时，首先要给模块起一个唯一的名字。程序块自动配置为无输入参数，一个输出参数(OK)。

模块结构的程序执行时，_MAIN 块自动执行。其他块由_MAIN 块，其他块或者自身的程序调用执行。下面例子中，%M00001 为 ON 时，执行 ProcessEGD 程序块：



程序块和本地数据

程序块支持全局数据%P。除_MAIN 块外的其他程序块都有自己的本地数据%L。程序块不会从调用他的块继承本地变量%L。

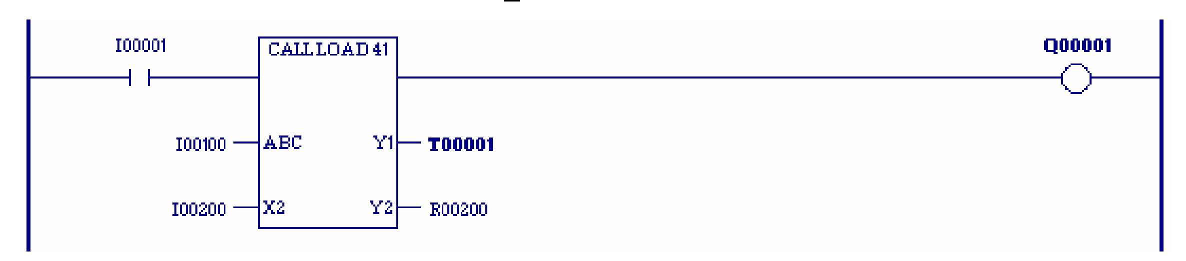
程序块的参数使用

每一个程序块自动定义 1 个正式的'电流'(或 OK)输出参数，名称为 Y0。Y0 为布尔型变量，长度为 1，指示初始的执行结果。如果程序块正常执行，此值为 1。可以用此程序块内的逻辑读写这一个布尔变量的状态。

参数化模块

除_MAIN 块外的其他程序块都可以是参数化模块。声明一个参数化模块时，必须指定一个唯一的模块名。参数化模块最多可以指定 63 个输入参数和 64 个输出参数。

可以通过_MAIN 块，其他程序块以及参数化模块自身的逻辑调用参数化模块。下面例子中，%I00001 为 ON 时，执行名为 LOAD_41 的参数化程序块



参数化程序块和本地数据

参数化程序块支持全局数据%P。参数化程序块没有自己的本地数据%L，但是可以从调用他的块继承本地变量%L，FST_EXE 系统变量和用于更新定时器函数的时间标签数据。如果参数化程序块中使用了%L 变量，并且这个块被_MAIN 块调用，%L 变量值将从%P 变量继承(例如%L0005 = %P0005)。

注意: 时用编程软件在线编辑参数化模块时，可能继承%L 数据而导致%L 数据超出允许的范围。是用逐字替换重新存储变量来纠正变量地址无法修复这种错误，因为变量仍然存放在变量表中。在变量表中删除变量时不会引起 CPU 更新，所以参数化模块人染会发现超出范围错误。要修复这种错误，需要先删除程序中的超范围变量，再删除未使用的变量。

如何使用参数化模块的参数

参数化模块可以定义 0~63 个输入参数和 1~64 个输出参数。电流输出(或 OK) 参数，名称为 Y0，这个参数是自动为每个参数化模块设定的。这个参数是一个布尔变量，指示参数化模块的执行状态。可以被参数化模块逻辑读写。

下表列举参数化模块的参数类型，长度和许可机制。(关于可以使用的参数类型的定义，见6-14 页“参数许可机制”)

类型	*长度*	*缺省参数许可机制*
BOOL	1 to 256	INPUTS: 根据数值
		OUTPUTS: 根据数值结果; 除了 Y0, Y0 由初始数值结果决定
BYTE	1 to 1024	INPUTS: 由变量决定
		OUTPUTS: 由变量决定
INT, UINT 和 WORD	1 to 512	INPUTS: 由变量决定
		OUTPUTS: 由变量决定
DINT, REAL 和 DWORD	1 to 256	INPUTS: 由变量决定
		OUTPUTS: 由变量决定
函数块*	1	INPUTS: 由变量决定
		OUTPUTS: 不允许

* TYPE 函数块最多有 16 个输入参数

PAC 系统缺省参数许可机制相当于 90-70 控制器的 PSB 参数。这种格式参数的许可机制不允许改变缺省值。

参数化模块执行时， 结果或实际参数读入参数化模块。一般来讲，给格式参数的结果可能来自任何存储类型，可能是数据流，也可能是常数(格式参数长度为 1 时) 下表阐述了一般规则下对结果的限制：

- %S 存储器不能赋给任何输出参数。因为不允许用户逻辑向%S 存储器写数据。
- 非直接变量作为结果时，在执行参数化模块调用前就将其解码，并将得到的直接变量输入参数块执行。例如，%R1 值为 10，@R1 作为调用的结果，在调用模块之前，将 @R1 解码为%R10，然后%R10 作为参数化模块的结果。在模块执行过程中，不管 %R1 的值是否变化，其值一直为%R10。

一般来讲，只要类型和长度与指令、函数或程序块调用要求兼容，参数化模块的格式参数可以用于任何指令，GE Fanuc 函数或任何程序块调用。下表列出了使用格式化参数的一般规则：

- 格式参数不能用作遗留转换结点或线圈以及 FAULT, NOFLT, HIALM 或 LOALM 结点。但是可以用作 IEC 转换结点和线圈。
- 布尔输入变量不能用于线圈，或作为 GE FANUC 函数或模块调用的输出。
- 格式参数不能用于 DO I/O 函数
- 格式参数不能和非直接变量一起使用

函数块

函数块是用户定义的具有参数和临时数据的逻辑块。用户可以定义自己的函数块，而不必局限于 PAC 系统指令设定的标准函数块。很多情况下，这个特性的使用可以使整个程序的变小。

成员变量不会做为参数输入函数块或者由函数块输出，成员变量只在函数块的逻辑中使用

一旦定义了函数块，可以生成很多个程序块实例。每个实例有自己唯一的函数块实例数据，这个事例中包含除含参变量外的所有输入输出参数。函数块在某个实例上被调用时，函数块逻辑在这个实例数据的实例拷贝上执行。实例数据的值从一次执行延续到下一次执行。

函数块的执行不能由中断触发。

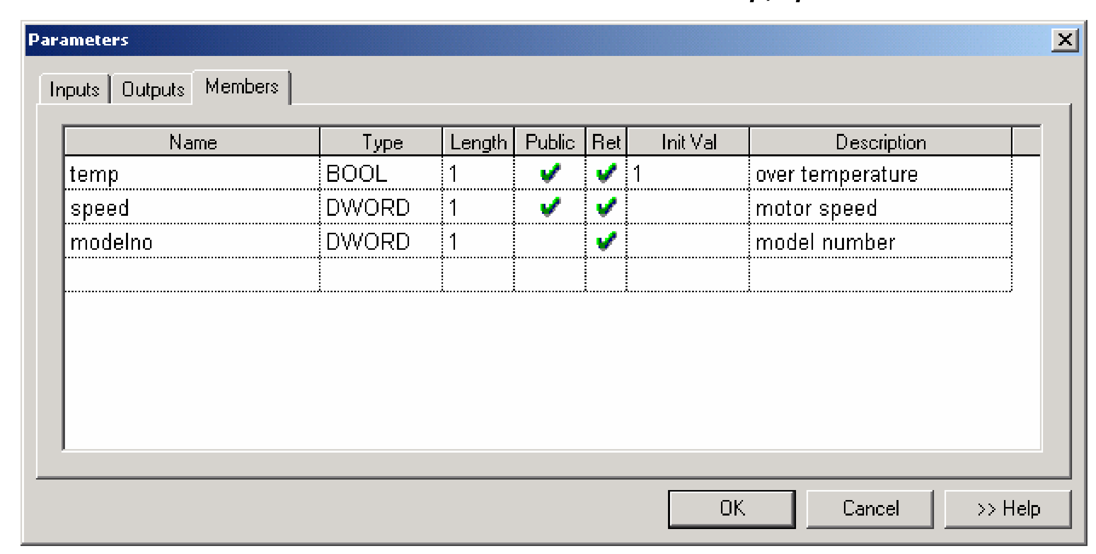
使用 LD 或者 ST 创建函数块逻辑。函数块逻辑可以对其他所有 PAC 块进行调用(程序块，参数化块，外部块和其他函数块)。LD 或 ST 内的程序块(程序块，参数化块，外部块和其他函数块)可以调用函数块。

除特殊情况外，PAC 系统用户自定义程序的执行遵循 IEC 61131-3 标准

定义一个函数块

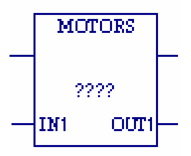
要在编程软件中创建一个函数块，首先在程序块文件夹中创建一个 LD 或 ST 块。在块的特性(Properties)栏中，选择函数块。

要为函数块定义实例数据，选择块属性的参数栏。输入输出参数定义方法与参数化模块的定义方式相同。下面的例子中，定义了 3 个内部成员变量: ***temp, speed 和 modelno.***

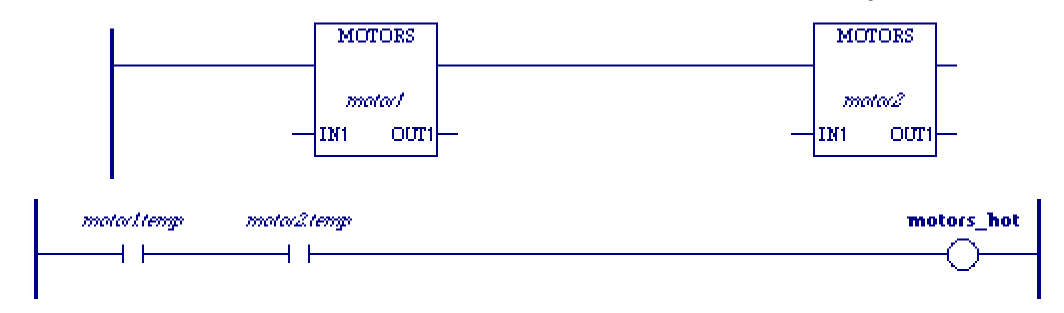


Creating Function Block Instances 创建函数实例

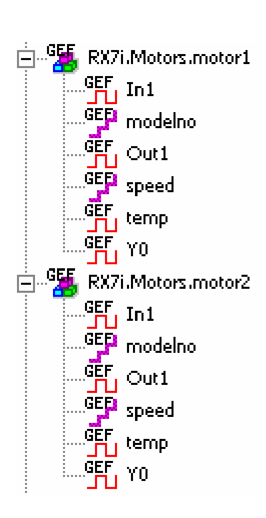
要创建一个函数块实例，首先在函数属性栏给此实例分配一个变量名，再在逻辑中调用即可(????)。



下面的例子中，为函数块 Motors 创建了两个实例，跟实例相关的两个实例变量为 motors.motor1 和 motors.motor2，第二行使用了两个实例的内部变量 ***temp***。



实例数据结构



每个函数的实例自动创建格式为 ***function_block_name.instance_name*** 的变量。实例数据的单个复合变量为*结构*类型。右侧的例子显示了 Motors 函数块的两个实例的变量结构。每一个实例变量又有对应 ***In1, Out1***, ***Y0*** 以及内部变量 ***modelno, speed 和 temp*** 的分量

和映射变量不同，实例是作为符号变量创建的。这确保实例数据对应变量名而不是对应存储器地址。如果对应存储器地址，函数块数据元素将不能有别名。非直接变量不能用作实例变量，因为符号变量不能使用非直接变量。

函数块与作用范围

和参数化子程序不同，函数块可以有自己的临时变量%L

缺省情况下，函数块的内部变量有自己的作用范围，这些变量只在函数块内部可见。不能使用外部逻辑和硬件配置对这些变量进行读写。要从函数块外部进行读写，必须将这些变量的作用范围改为全局作用。内部变量作用范围改为全局后，函数块外的逻辑仍然只能读取这些变量的数据而不能改变这些变量的值。

注意: 如果将内部变量作用与改为全局作用，你的应用程序将不满足 IEC 的要求。

函数块的参数使用

函数块最多支持 63 个输入，64 个输出。

每一个函数块都有一个预先定义的输出参数,Y0,CPU 在每次执行模块实现将其置 1。可以通过内部逻辑控制 Y0 值,并且指示块的输出状态。

下表列出了函数块参数的类型，长度和参数许可机制。关于参数许可的其他信息，参考 6-14 页，" 参数许可机制" 。

<table>
<tr><th>类型</th><th>长度</th><th>参数许可机制</th><th>实例数据参数的保持</th></tr>
<tr><td rowspan="2">BOOL</td><td rowspan="2">1 to 256</td><td>INPUTS:根据变量，数值或者数值结果.
(缺省: 数值)</td><td>因为不存储在实例数据中，所以不适用于根据变量的情况
对于数值和数值结果，可以是保持型的，也可以是非保持型的</td></tr>
<tr><td>OUTPUTS: 根据结果; 除了 Y0, Y0 由初始值结果决定</td><td>保持型的(缺省值)或者非保持型的</td></tr>
<tr><td rowspan="2">BYTE</td><td rowspan="2">1 to 1024</td><td>INPUTS: 根据变量，数值或者数值结果.
(缺省: 数值)</td><td rowspan="2">对于数值和数值结果，保持型的
不适用于变量</td></tr>
<tr><td>OUTPUTS: 根据结果</td></tr>
<tr><td rowspan="2">INT, UINT 和 WORD</td><td rowspan="2">1 to 512</td><td>INPUTS: 根据变量，数值或者数值结果.
(缺省: 数值)</td><td rowspan="2">对于数值和数值结果，保持型的
不适用于变量</td></tr>
<tr><td>OUTPUTS: 根据结果</td></tr>
<tr><td rowspan="2">DINT, REAL 和 DWORD</td><td rowspan="2">1 to 256</td><td>INPUTS: 根据变量，数值或者数值结果.
(缺省: 数值)</td><td rowspan="2">对于数值和数值结果，保持型的
不适用于变量</td></tr>
<tr><td>OUTPUTS: 根据结果</td></tr>
<tr><td rowspan="2">函数块*</td><td rowspan="2">1</td><td>INPUTS: 根据变量</td><td rowspan="2">由于是根据变量的，所以不适用</td></tr>
<tr><td>OUTPUTS: 不允许</td></tr>
</table>

* TYPE 函数块最多 16 个输入参数

如果输入参数是根据变量或根据数值结果的，那麽需要一个输出。函数块所有其他参数可选。即，不是函数块的每个实例都要有结果。如果莫个可选参数没有结果，还参数相关的变量保持当前值。

函数块输出不能作为结果传给根据变量或者根据数值结果的输入参数。这个限制阻止对函数块输出的修改。

函数块内部成员变量的使用

函数块可以有任意数量的内部成员变量。内部成员变量不作为输入输出参数。内部成员变量不能和函数块定义的参数同名。

内部变量可以为：

- PAC 系统支持的任何基本类型(BOOL, INT, UINT, DINT, REAL, BYTE, WORD 和 DWORD)：
- 函数块类型。这些成员变量作为嵌套实例。例如，函数块"Motor"可以有一个"Valve,"类型的内部变量。注意，定义一个函数块类型的成员变量并不会创建一个实例。

  嵌套实例不能和定义的函数块类型相同，因为这会导致无穷的回归定义。任意一级的嵌套实例也不能和其父块类型相同。例如，如果 Valve 函数块有一个"Motor"类型子块，那麽函数块 Motor 不能有 Valve 类型的内部变量
- 一个一维数组

布尔型的内部变量可以是保持型的(缺省值)，也可以是非保持型的。其他类型的变量必须是保持型的。

对应于函数块输入参数的成员变量不能从函数块外部进行读写(比 IEC 61131-3 的要求更加严格)。对应于函数块输出参数的成员变量能从函数块外部读取

可以为具有基本类型的内部成员变量赋初值。同样的初始值适用于函数块实例。给定初始值后，在第一次转换到运行状态时，内部成员变量值设为 0。

嵌套实例的内部成员变量值由函数块类型定义赋初值。

运行模式存储时，初始值不存储。初始值只在停止模式存储时起作用。

函数块逻辑

布尔参数或内部变量实例可以强制为 on 或 off 状态，或者用于状态转换检测指令。布尔型输入参数通过参数使用时不能被强制或被 90-70 系列状态转换检测指令(POSCOIL, NEGCOIL, POSCON and NEGCON)使用，因为他们的值不是存储在实例数据中。

函数块的所有输入参数和他们对应的实例数据元素可以被函数块的逻辑读取。

通过变量或数据结果传到函数块的输入参数可以被函数块逻辑更改。通过数值传到函数块的输入参数不能被函数块逻辑更改。注意，对通过数值传到函数块的输入参数的更改的限制不适用于其它类型的块。

函数块的所有输出参数可以被块内逻辑修改。

其他块对函数块的操作

全局作用范围的函数块实例可以有其他函数块，或任何其他类型的块的逻辑激活。

作为结果传送给函数块的函数块实例可以被这个函数块的逻辑激活

作为结果传送给参数化块的函数块实例可以被这个参数化块的逻辑激活

输出参数和对应的作为结果传送的函数块的实例数据元素，可以被接收块的逻辑读取，但不能修改。作为结果传送的函数块的输入参数，不可以被接收块的逻辑读写。作为结果传送的内部变量不能被接收模块的逻辑修改，如果这些参数使全局的就可以读，否则不能读取。

外部块

外部块是用外部开发工具，比如 PAC 的 C 程序开发工具，开发的。关于外部块的细节，参考 PAC 系统的 C 语言编程工具包用户手册 GFK-2259。

除_MAIN 块外的其他程序块都可以是外部模块。声明一个外部块时，必须指定一个唯一的模块名。外部块最多可以指定 63 个输入参数和 64 个输出参数。

可以通过_MAIN 块，其他程序块，函数块以及参数化模块的逻辑调用外部块。外部块本身不能调用其他模块。下面例子中，%I00001 为 ON 时，执行名为 EXT_11 的外部块。

I00001
CALL EXT 11
Q00001
I00100 X1 Y1 T00001
I00200 X2 Y2 R00200

注意: 其他类型的块不同的是，外部块不能调用其他块。

外部块和本地数据

外部块可以使用全局变量%P，不支持本地变量%L，但可以从调用块中继承本地变量%L。外部块还从调用他的块中继承系统变量 FST_EXE 和用于更新定时器函数块的"时间标签"。如果参数化程序块中使用了%L 变量，并且这个块被_MAIN 块调用，%L 变量值将从%P 变量继承(例如%L0005 = %P0005)。

C 变量初始化

外部块存储到 CPU 之后，他的全局变量和静态变量的副本存储起来。然而，如果外部块声明静态变量时没有赋初值，则初始值必须在 C 程序中给定。(参考 PAC 系统的 C 语言编程工具包用户手册 GFK-2259 内的“全局变量初始化”和“静态变量”) 在从停止状态转换到运行状态时，会用保存起来的初始值重新初始化块内的全局变量和静态变量。

外部块的参数使用

每一个外部块可以定义 0~63 个输入参数和 1~64 个输出参数。电流输出(或 OK) 参数，名称为 Y0，这个参数是自动为每个外部块设定的。这个参数是一个布尔变量，指示外部块的执行状态。可以被外部块逻辑读写。

The following table gives the TYPEs, LENGTHs, and parameter passing mechanisms allowed for external block parameters. 下表列出了外部块参数的类型，长度和参数许可机制。

类型	*长度*	*缺省参数许可机制*
BOOL	1~56	INPUTS: 根据变量
		OUTPUTS: 根据数值结果; 除了 Y0, Y0 由初始数值结果决定
BYTE	1~024	INPUTS: 根据变量
		OUTPUTS: 根据变量
INT, UINT 和 WORD	1~12	INPUTS: 根据变量
		OUTPUTS: 根据变量
DINT, REAL 和 DWORD	1~56	INPUTS: 根据变量
		OUTPUTS: 根据变量

PAC 系统缺省参数许可机制和 90-70 控制器的许可机制一致。这时，格式参数的许可机制只能是缺省状态，不能改变。

必须为每一个输入参数和每一个输出参数定义一个名字。参数长度为 1~3 个字符，以便于将参数名显示在调用这个外部块的指令上。

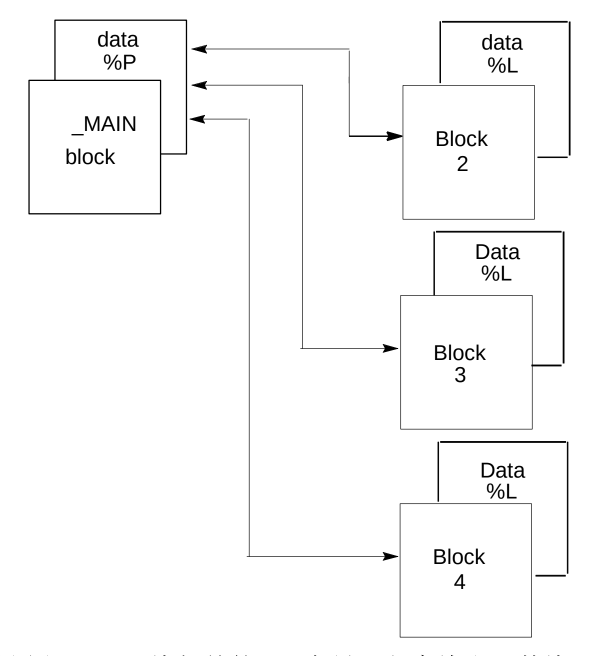
结果或实际参数在执行调用时进入这个外部块

结果可以是包含非直接变量，数据流或者常量 (如果对应的参数长度为 1)变量地址。

Local Data

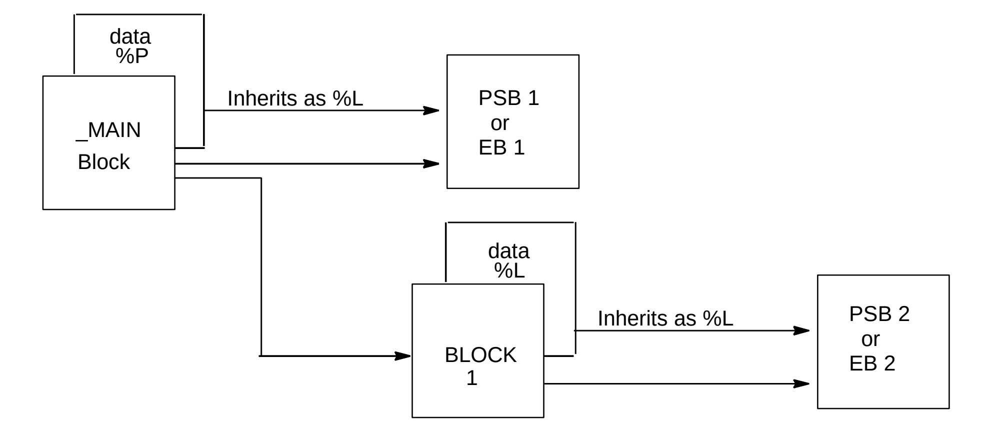
块结构的每一个程序块或者函数块有一个相关的本地数据块。_MAIN 数据块存储器使用%P 变量，所有其它数据块存储器使用%L 变量。

数据块的大小依赖于块内%L 变量的最大范围以及所有块的%P 变量的范围。



程序中所有块可以使用和_MAIN 块相关的%P 变量。程序块和函数块可以使用自己的%L 数据和所有块都可以使用的%P 数据。_MAIN 块不可以使用%L 变量。

外部块和参数化模块可以使用调用他们的块的%L 数据，也可以使用_MAIN 块的%P 数据。如果外部块和参数化模块被_MAIN 块调用，则外部块和参数化模块使用的%L 变量对应的转化为对应的%P 变量(例如, %L0005 = %P0005)。除了继承调用块的本地数据，外部块和参数化模块还从调用块中继承 FST_EXE 状态。



参数许可机制

所有块(_MAIN 块除外)都至少有 1 个参数，所以都受参数许可机制的影响。参数许可机制描述数据如何从调用块运行结果进入到被调用块中，以及参数如何从被调用块进入到调用块的结果中。

PAC 系统支持 5 种参数许可机制：通过变量，通过数值，通过数值结果，通过结果和通过初始化值的结果。由类型，长度和许可机制具体确定一个参数。

当参数是**通过变量的，**变量地址进入被调用块。被调用块内读写这个参数的所有逻辑直接访问实际变量。

当参数是**通过数值的，**变量值拷贝到被调用块的本地堆栈寄存器中。被调用块内读写这个参数的所有逻辑直接访问堆栈寄存器。

当参数是**通过数值结果的，**变量值拷贝到被调用块的本地堆栈寄存器中，变量地址被保存。被调用块内读写这个参数的所有逻辑直接访问堆栈寄存器。被调用块执行完成后，堆栈寄存器内的数值重新拷贝到实际的变量地址中。在被调用块执行过程中，实际变量值没有发生变化，但是被调用块执行完成后，实际变量值被更新。

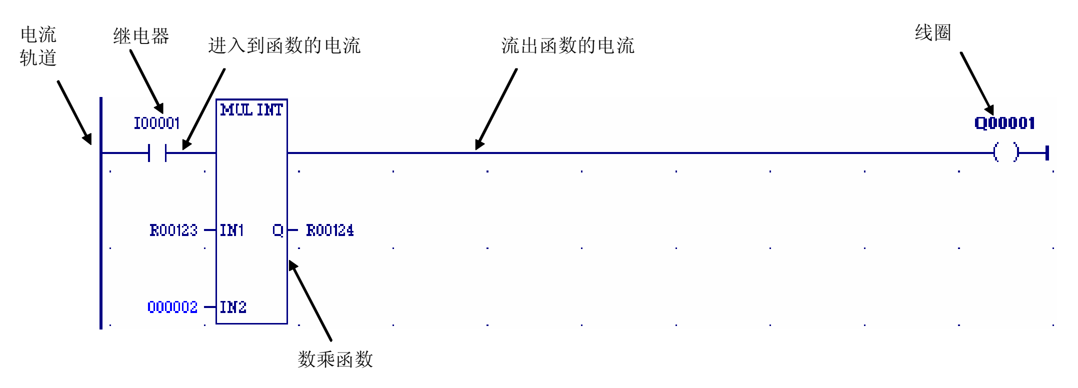
当参数是**通过结果的，**会在被调用块的本地堆栈寄存器中分配一个空间，变量地址被保存。被调用块内读写这个参数的所有逻辑直接访问堆栈寄存器。被调用块执行完成后，堆栈寄存器内的数值重新拷贝到实际的变量地址中。在被调用块执行过程中，实际变量值没有发生变化，但是被调用块执行完成后，实际变量值被更新。

当参数是**通过初始化值的结果的，**会将 TRUE 的初始化值拷贝到被调用块的本地堆栈寄存器中，变量地址被保存。被调用块内读写这个参数的所有逻辑直接访问堆栈寄存器。被调用块执行完成后，堆栈寄存器内的数值重新拷贝到实际的变量地址中。在被调用块执行过程中，实际变量值没有发生变化，但是被调用块执行完成后，实际变量值被更新。所有块的 OK 输出参数使用**通过初始化值的结果的**许可机制

语言

梯形图(LD)

用梯形图语言书写的逻辑中包含一系列的回路，这些回路从上至下依次执行。将逻辑的执行想象为 ‘电流’，电流沿着左侧的 ‘轨道’向下，并依次从左向右执行遇到的每个回路。



逻辑电流通过一系列由简单程序指令控制的回路，工作原理如同机械继电器和输出线圈。一个继电器是否通过逻辑电流依赖于和继电器相关的存储器的内容。例如，如果相关存储器内容为 1，则通过逻辑电流，为 0 则不通过逻辑电流。

通常一个指令接收到反向电流指令时，本指令不执行，并且将反向电流传向回路中的下一个指令。然而，定时器或计数器等指令受到反向电流时也会执行，并且向后传播反向电流。一旦一个回路执行完以后，不管输出正向还是反向电流，电流都回传给下一回路。

在一个回路中，有许多复杂的函数是标准 GE Fanuc 函数库的函数。这些函数可以完成数据移动功能，数学计算和 cpu 与系统内其他设备的通讯控制等功能。一些程序函数，如跳转(Jump)函数和主控器控制延时(Master Control Relay)函数，可以控制程序自身的执行。总之，大量的梯形图指令和标准 GE Fanuc 库函数组成了 CPU 的指令设置。

注意: 90-70 系列 PLC 按列执行：他们从上到下执行第一列，之后向右执行第二列，直到执行完所有列。PAC 系统和 90-30 系列 CPU 是按行执行的：他们从第一行开始执行，之后依次向下执行第二行，第三行…..直到执行完所有程序。例如，参考附录 C” 列为主与行为主的梯形图逻辑的执行”

结构化文本

结构化文本(ST)编程语言 IEC 1131-3 文本化语言。结构化文本程序包括一系列的声明，这些状态由表达式和语言关键字创建。一个声明令 PLC 做一个指定的动作。声明提供了变量分配，条件评估，迭代和调用函数和函数块功能。关于 PAC 系统支持的 ST 的声明，参数，关键字和操作者，参考第十二章“结构化文本”

程序块，参数化程序块和函数块可以用 ST 编写。_MAIN 程序块也可以用 ST 编写。

用 ST 编写的块可以调用程序块，参数化程序块和函数块。

控制程序的执行

为了满足系统的时间要求，有许多方式可以控制程序的执行。PAC 系统 CPU 中有几种强力的控制函数，可以将这几个函数写入应用程序中来改变 CPU 执行程序和扫描 I/O 的方式。关于如何使用这些函数，详见第九章和第十章。

下面是一般使用方法的部分列表：

- 跳转(JUMPN)函数用于将程序执行位置向前或向后跳转。JUMPN 函数激活时，被跳过的程序中的线圈保持之前的状态(接到反向电流，如同执行了主控器控制延时)。跳转不能跨程序块进行。
- 嵌套的主控器控制延时(MCRN)函数用于将反向电流赋给程序逻辑的某一部分。逻辑向前执行，这部分程序中的线圈收到反向电流。主控器控制延时最多可嵌套 255 层。
- 延缓 I/O 函数停止本周期的输入输出扫描。如果需要的话，可以在程序中使用 DO I/O 指令进行 I/O 更新。
- 服务请求指令用于延缓或改变分配个窗口的扫描端口的时间
- 程序可以根据重要性和时间限制来增加或减少模块的调用次数。调用函数用于激活某段程序或某个模块的执行。调用函数之前的条件逻辑用于控制在什麼情况下让 CPU 执行程序块逻辑。程序块执行完成后，程序从调用指令点之后继续执行。
-

中断驱动块

3 种类型的中断用于启动程序块：

- 由 CPU 产生，中断的时间间隔由用户指定。如果设定了初始延时，则在完成停止-运行转并经过初始延时时间后，开始计算中断时间。
- I/O 中断由 I/O 模块发出，I/O 中断用于指示离散输入状态的改变(上升沿/下降沿)，模拟量范围限制(高/低报警)和高速计数器事件。
- 模块中断由 VME 模块产生。每个模块支持单个中断。

注意

中断驱动块执行后能中断非中断驱动块的执行。如果正在中断的逻辑和已中断的逻辑访问相同的数据，会产生不可预期的错误。需要的话，在共享数据正在被访问时，服务请求程序 17 或服务请求程序 32 可以临时屏蔽 I/O 和定时中断程序的执行

中断处理

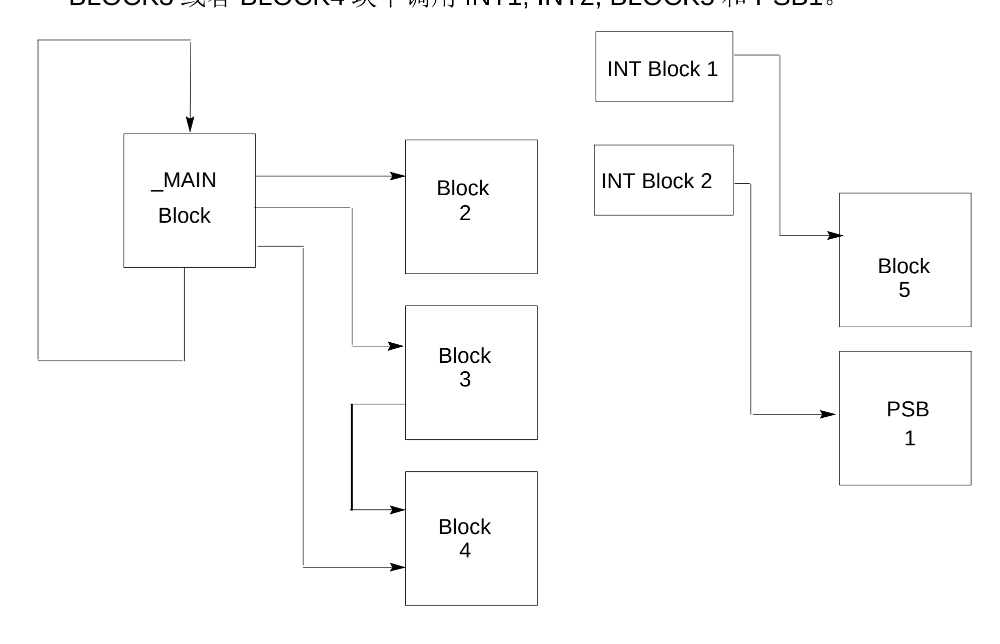
只要程序块除了 OK 输出外没有其他的参数，I/O、模块或者定时中断可以用于_MAIN 块外的任何程序块。在中断和程序块关联后，每次中断触发后，都会执行这个程序块。一个模块可以和多个定时中断、I/O 中断以及模块中断相关联。关于程序块中断的优先级，参考 6-20 页"中断编制"。

如果参数化程序块和外部程序块被中断触发，他将%P 数据作为本地数据%L 继承。例如，参数化程序块和外部程序块内的本地变量%L00005 继承的是变量%P00005。

注意: 作为中断结果运行的程序块内有定时器函数块时，这个定时器块的时间不累加。

中断触发的程序块可以调用其他的程序块。中断驱动执行时的应用程序堆栈不同于正常的模块化程序执行时的堆栈。特别的，嵌套调用限制不同于从_MAIN 块调用的情况。如果一个调用导致堆栈空间不足，无法完成调用，CPU 会记录为“堆栈溢出故障”。

注意: GE Fanuc 建议不要在_MAIN 块中使用，因为中断驱动块和非中断驱动块可能在不确定的时间同时读写相关的全局数据。下面的例子中不应该在_MAIN BLOCK2, BLOCK3 或者 BLOCK4 块中调用 INT1, INT2, BLOCK5 和 PSB1。



定时中断

某个程序块可能要在上电后定期执行：

要配置定时中断块，需要在程序块属性栏中设定以下参数：

时间基准 (Time Base)	设定的时间间隔和延时的最小时间单元。时间基准可以设定为 1 秒，0.1 秒和 0.001 秒
时间间隔 (Interval)	设定值*时间基准后，得到的值为程序块两次执行的时间间隔
延时(Delay)	(可选择)设定值*时间基准后，得到的值为延时多长时间开始第一次执行程序块

定时中断块第一次执行的时间为：
CPU 进入运行模式后((延时*时间基准) + (时间间隔*时间基准))

I/O 中断

程序块可以由某些硬件输入的中断信号触发。例如，24 VDC，32 回路的输入模块(IC697MDL650)，不管输入信号是上升沿还是下降沿，第一次输入时都发出一个中断。如果模块配置中将中断功能使能，则中断可以触发执行一个程序块。

I/O 中断在程序块属性栏中配置，触发必须使用全局变量%I, %AI 或者%AQ。

模块中断

如果在模块硬件配置中将 VME 中断参数使能，程序块可以由 VME 模块输入的中断信号触发。PAC 系统 CPU 允许每个模块有一个中断。

要配置模块中断，在程序块属性栏中标明模块所在的机架/槽号/中断 ID 即可。

中断块时序

你可以在目标层级选择 1 种或者 2 种类型的中断程序块：

- 正常程序块时序允许有最多 64 个 I/O 和模块中断以及 16 个定时中断。正常程序块时序情况下，所有的中断触发块有相同的优先级。这时缺省的时序模式。
- 具有优先级的程序块时序允许有最多 32 个中断触发。具有优先级的程序块时序情况下，为每一个触发指定一个优先级。

正常程序块时序

中断驱动逻辑在系统内所有的用户程序中具有最高的优先级。CPU 优先执行中断触发的程序块。在中断驱动的程序块执行完毕后，CPU 再接着执行其他程序。

如果正在执行中断块时收到其他的中断指令，CPU 会将这些指令放到完成本次中断后需要执行的任务队列中。定时中断驱动程序块排在 I/O 驱动程序块或模块驱动程序块前面。I/O 驱动程序块或模块驱动程序块按收到的顺序排队。如果中断驱动块已经在队列中了，调用程序块的其他中断会被屏蔽掉。

优先级程序块时序

优先级时序允许你为每一个触发指定一个优先级。优先级数值范围为 1~16，1 优先级最高。单个模块可以有多个优先级不同的中断，也可以有多个优先级相同的中断。

将接收到的中断级别和正在执行的程序块的中断级别进行比较后，会按如下方式执行程序块：

- 如果接收到的中断级别高于正在执行的程序块的中断级别，则停止执行当前程序块，转而执行和新收到的中断相关的程序块。
- 如果接收到的中断级别等同于正在执行的程序块的中断级别，则继续执行当前程序块。将新收到的中断放入队列中。
- 如果接收到的中断级别低于正在执行的程序块的中断级别，则将新收到的中断放入队列中。

CPU 执行完一个中断程序块后，将执行和队列中级别最高的中断相关联的程序块。如果 CPU 在执行某个中断程序块的过程中接收到更高级别的中断，则在执行完和该中断相关的程序块后，继续执行未执行完的那个程序块。

如果队列中的多个程序块有相同的中断优先级，则执行顺序不确定。

注意: 某些函数，如 DOIO, BUS_RD, BUS_WRT, COMMREQ 和某些 SVC_REQ 函数会产生一些具有相同优先级的其他队列程序块。

Chapter 7 程序数据

本章描述了应用程序中可以使用的数据类型，以及这些数据如何在 PAC CPU 存储器中存储。

- 变量
- 变量存储器
- 用户变量大小和缺省值
- 网全局数据
- 转换和覆盖
- 逻辑和数据的保持
- 数据范围
- 系统状态变量
- 程序函数如何处理数字数据
- 逐字替换

变量

变量是已命名的存储数据值的存储空间。他代表了目标 PAC CPU 内的存储位置。

变量可以映射到变量地址(例如, %R00001)，如果没有将变量映射到变量地址，则将这个变量看作符号变量。编程软件将符号变量映射到 PAC 系统用户存储空间的某个部分。

变量能存储的值依赖于他的数据类型。例如，UNIT 数据类型存储无符号整数，没有小数部分。数据类型在 7-16 页“程序函数如何处理数值数据”

编程软件中，工程的所有变量显示在浏览器的变量键下。以可以在变量键下创建，编辑和删除变量。一些变量由某些部分自动创建(如在梯形图逻辑中增加定时器指令时，就会自动添加定时器变量)变量类型和地址等其他属性，在 Inspector 中配置。

在编程软件中创建对象时，会自动创建系统变量，关于系统变量的更多信息，参考 7-11 页

映射变量

映射变量(手动定位)有一个确定的变量地址。详见 7-4 页，变量存储器类型及使用。

符号变量

符号变量是没有分配确定地址的变量(与典型高级语言的变量类似) 。除了这部分所述的，你可以象使用映射变量一样使用符号变量。

在编程软件中，符号变量的地址栏空置。在变量属性栏中删除变量地址就可以将映射变量转化为符号变量。同样的，在符号变量的变量地址栏输入地址就可以将符号变量转换为映射变量。

符号变量所需的存储空间根据用户空间计算。为这些变量预留的空间大小在 CPU 硬件配置下的存储器键内配置。

对符号变量使用的限制

- 符号变量不能用作非直接变量(例如，@变量名)(详见 7-4 页，非直接变量描述)
- 符号变量不能用于 EGD 页
- C 块不支持符号变量
- 符号变量不能用于 COMMREQ 状态字
- 符号变量不能作为字内的位用于结点或线圈
- WEB 页不支持使用符号变量
- 符号变量不能用作硬件模块的 I/O 点，状态字等等。
- 符号型布尔变量不允许用作非布尔参数

变量存储器

CPU 以位存储器和字存储器的方式存储程序数据。以不同的特性将两种类型的存储器分解成不同的类型。如下面叙述的，每一种类型的存储器一般用于特定类型的数据。然而，实际存储器的分配要更有弹性。

存储定位以文字标识符(变量)作为索引.变量的字符前缀确定存储区域。数字值是存储器区域的偏移量，例如%AQ0056

字(寄存器)变量

类型	描述
%AI	前缀%AI 代表模拟量输入寄存器。模拟量输入寄存器保存模拟量输入值或者其他的非离散值。
%AQ	前缀%AQ 代表模拟量输出寄存器。模拟量输出寄存器保存模拟量输出值或者其他的非离散值。
%R	前缀%R 代表系统寄存器变量。系统寄存器保存程序数据比如计算结果。
%W	保持型的海量存储区域，变量为%W (字存储器)类型
%P*	前缀%P 代表程序寄存器变量。在_MAIN 块中存储程序数据。这些数据可以从所有程序块中访问。%P 数据块的大小取决于所有块的最高的%P 变量值。%P 地址只在 LD 程序中可用，包括 LD 块中调用的 C 块，P 变量不是整个系统范围内可用的。

注意: 所有的寄存器变量在掉电/上电后，仍然保持原来的数据。

非直接变量

非直接变量允许你将分配给 LD 指令的操作数作为一个指针指向其它数据，而不是作为实际的数据。非直接变量值用于字存储器区域(%R, %W, %AI, %AQ, %P 和%L)。需要两个%W 变量指引%W 中的非直接变量的位置。例如，@%W0001 需要%W2:W1 作为双字索引指示%W 存储器范围。需要双字索引，是因为%W 的大小超过 65K。

符号变量不能用作非直接变量

要指定非直接变量，先敲入@，再敲入变量地址或变量名，例如，如果%R00101 的值为 1000，则@R00101 使用的是%R01000 内包含的值。

对很多字寄存器使用同样的操作时，非直接地址非常有用。非直接地址的使用还能避免在应用程序中重复使用梯形图逻辑。也可以用在循环的情况，寄存器每次加一个常数或某一个设定值，直到加到最大值为止。

字变量中的位

字变量中的位允许你设定字的某一位的值，可以将这一位做为二进制表达式输入输出以及函数和调用的位参数(例如 PSB)。这个特征只适用于保持型存储器的位变量。自结构的变量的位号必须为常数。

你可以使用编程器或者 HMI 将字中的某一位设定为 ON 或 OFF,也可以监控这一位。C 块也可以对字中的某一位进行读取，更改和写入操作。

字变量中的位可以在以下情况时使用：

- 保持型 16 位存储器(AI, AQ, R, W, P, and L)
- 除了状态转换结点(POSCON/NEGCON)和状态转换线圈(POSCOIL/NEGCOIL)外的所有结点和线圈。
- 在所有函数和接受单个的或者非捆绑位的调用参数

接受非捆绑离散变量的函数	*参数*
ARRAY MOVE (BIT)	SR and DS
ARRAY RANGE (BIT)	Q
MOVE (BIT)	IN and Q
SHFR (BIT)	IN, ST and Q

字变量中的位的使用限制：

- 字变量中的位不能用于状态转换结点(POSCON/NEGCON)和状态转换线圈(POSCON/NEGCON)
- 位号(索引)必须为常数，不能为变量。
- 不支持以常数作为位地址
- 非直接变量不能用作 16 位存储器的位地址
- 不能在 16 位存储器中强制一位

例子:

%R2.X [0] 表示 %R2 的第 1 位 (最低位)。

%R2.X [1] 表示 %R2 的第 2 位.

例子中[0] 和[1]是位索引。不同类型变量的位索引范围分别为：

BYTE 变量	[0]~ [7]
WORD, INT 或者 UINT 变量	[0]~[15]
DWORD 或者 DINT 变量	[0]~ [31]

位(离散)变量

类型	*描述*
%I	代表输入变量。%I 变量位于输入状态表中，输入状态表中存储了最后一次输入扫描过程中输入模块传来的数据。用编程软件为离散输入模块指定输入地址。地址指定之前，无法读取输入数据。%I 寄存器是保持型的。
%Q	代表自身的输出变量。线圈检查功能核对线圈是否在延时线圈和函数输出上多处使用。你可以选择线圈检查的等级。(Single, Warn Multiple, or Multiple) %Q 变量位于输出状态表中，输出状态表中存储了应用程序对最后一次设定的输出变量值。输出变量表中的值会在本次扫描完成后传送给输出模块。用编程软件为离散输出模块指定变量地址。地址指定之前，无法向模块输出数据。%Q 变量可能是保持型的，也可能是非保持型的。
%M	代表内部变量。线圈检查功能核对线圈是否在延时线圈和函数输出上多处使用。%M 变量可能是保持型的，也可能是非保持型的。
%T	代表临时变量。线圈检查功能不会核对线圈是否多处使用，因而即使使用了线圈检查功能，也可以多次使用%T 变量线圈。当然我们建议不要这样使用，因为这样做会更难查错。在使用剪切/粘贴功能以及文件写入/包含功能时，%T 的使用会避免产生线圈冲突。因为这个存储器倾向于临时使用，所以在停止-运行转换时会将%T 数据清除掉，所以%T 变量不能用作保持型线圈。
%S %SA %SB %SC	代表系统状态变量。这些变量用于访问特殊的 CPU 数据，比如说定时器，扫描信息和故障信息。%SC0012 位用于检查 CPU 故障表状态。一旦这一位被一个错误设为 ON，在本次扫描完成之前，不会将其复位。 ■ %S, %SA, %SB 和%SC 可以用于任何结点。 ■ %SA, %SB 和%SC 可以用于保持型线圈 -(M)-. **注意:** 尽管编程软件强制逻辑在保持型线圈上使用%SA, %SB 和%SC 变量，大部分这些变量不会在有电池做后备电源的掉电/上电过程后保持原来的数据。 %S 可以作为字或者位串输入到函数或函数块。 %SA, %SB 和%SC 可以作为字或者位串输入，或从函数和函数块输出。 关于每一位的特性的详细描述，见 7-11 页"系统状态变量"
%G	代表全局数据变量。这些变量用于几个系统之间的共享数据的访问。

注意: 关于数据的保持，参考 7-19 页“逻辑和数据的保持”

用户变量范围和缺省值

用户变量的最大范围和缺省范围见下表

项目	*范围*	*缺省值*
点变量		
%I 变量	32768 位	32768 位
%Q 变量	32768 位	32768 位
%M 变量	32768 位	32768 位
%S (S, SA, SB, SC) 变量总计	512 位 (每个 128 位)	512 位 (128 each)
%T 变量	1024 位	1024 位
%G	7680 点	7680 点
变量总点数	107520	107520
字变量		
%AI 变量	0—32640 字	64 字
%AQ 变量	0—32640 字	64 字
%R, 1K word increments	0—32640 字	1024 字
%W	0—最大至用户 RAM 上限	0 字
变量字数总和	0—最大至用户 RAM 上限	1152 字
%L (每个块)	8192 字	8192 字
%P (每个程序)	8192 字	8192 字
Symbolic Memory		
离散符号变量	0—20,971,520 (位)	32768
非离散符号变量	0—5,242,880 字)	65536
符号变量总和	0—10,485,760 字节 (上述为符号变量所有可用存储的总和。其中包含其他用户存储器，程序等等)	149760

用户变量%G 和 CPU 存储器位置

CPU 有一个存储所有全局数据变量(%G)的数据空间。CPU 内存为这些为保留的空间大小为 7680 位。对于 90-70 系列的系统来说，编程软件将这个空间分为%G, %GA, %GB, %GC, %GD 和%GE 前缀的 6 个部分，每一个前缀的变量偏移量为 1-1280。PAC 系统将这些范围转换给%G 前缀的变量。

Genius 全局数据

PAC 系统支持多个控制系统在一个 Genius 网上共享数据。这种机制提供了一种自动并且重复传送%G, %I, %Q, %AI, %AQ 和%R 数据的方法。不需要为全局数据编写特别的程序，因为已将其集成到 I/O 扫描中了。所有有 Genius I/O 能力的 GE Fanuc 设备都可以向 RX7i 发送全局数据，也都可以从 RX7i 接收全局数据。

转换和覆盖

用户变量%I, %Q, %M 和 %G 以及布尔型的符号变量有转换和覆盖位。%T, %S, %SA, %SB 和 %SC 变量有转换位，但是没有覆盖位。CPU 将转换位用于计数器，状态转换结点和状态转换线圈。注意：计数器使用的转换位与结点和线圈使用的转换位不同。计数器使用的转换位存储在本地变两种。

变量的转换位指示了最近写入变量的值(ON, OFF)和之前写入变量的值是否相同。当两个值相同时，转换位 OFF。不同时，转换位 ON。每一次向变量写入数据时，转换位的值都会受影响。详变量写入值的数据源是非实质性的，可能是线圈的执行结果，已经执行了的函数的输出以及输入扫描后的变量存储器的更新等等。

设定了覆盖位以后，程序和输入设备不能更改该变量的值，该值只能通过编程器命令修改。覆盖不会保护转换位。如果尝试对覆盖存储器位置进行修改，则会将转换位清 0。

逻辑和数据的保持

Data is defined as retentive if it is saved by the CPU when the CPU transitions from STOP mode to RUN mode. 如果 CPU 在从停止模式转换到运行模式时保存了某个数据，则该数据被定义为保持型：

下列项目是保持型的：

- 程序逻辑
- 故障表和自诊断信息
- 程序逻辑的检查信息
- 覆盖和输出强制
- 字数据(%R, %W, %L, %P, %AI, %AQ)
- 位数据(%I, %G, 故障位置变量和保留位)
- 配置为保持型的%Q 和 %M 变量(%T 是非保持型的，所以在停止-运行转换过程中不会被保存)
- 非布尔型的符号变量
- 配置为保持型的符号变量
- 以电池为后备电源的 CPU 掉电上电过程中会将保持型的数据预先保存起来。但是故障位置变量和多数的%S, %SA, %SB 和%SC 变量不会保存。不过电池状态如何，CPU 会在上电时将这些值置 0。(见 7-11 页，系统状态变量特性描述)

将 %Q 和 %M 变量配置为保持型的时候，这些变量的值会在掉电或运行-停止-运行过程中保持下来。

数据范围

每一种用户变量都有自己的作用范围。即：它可能在整个系统中可用，对所有程序可用，只能用于某个程序或者只能在某个程序块中使用。

用户变量类型	*范围*	*作用范围*
%I, %Q, %M, %T, %S, %SA, %SB, %SC, %G, %R, %W, %AI, %AQ 有效变量，故障位置变量	全局	可以从任何程序，程序块或者主机访问。这些变量的缺省作用范围为用于整个系统(全局)，本地变量也可以使用这些种类的寄存器
符号变量	全局	可以从任何程序，程序块或者主机访问。符号变量的缺省作用范围为用于整个系统(全局)。本地变量也可以使用这符号型变量
%P	程序	可以从任何程序块访问，不可以从其他程序访问
%L	本地	只允许在程序块内访问(可使用主机访问)

在 LD 块内：

- %P 用作程序变量，可与其他程序块共享
- %L 是本地变量，只限于在本程序块内使用
- %I, %Q, %M, %T, %S, %SA, %SB, %SC, %G, %R, %W, %AI 和%AQ 变量在整个系统范围内可用。

系统状态变量

CPU 的系统状态变量为%S, %SA, %SB 和 %SC 变量。4 种定时结点包括#T_10MS, #T_100MS, #T_SEC 和 #T_MIN。系统状态变量的其他例子包括#FST_SCN, #ALW_ON 和#ALW_OFF

注意: %S 位是只读位；不要向这些位写数据。可以向%SA, %SB 和%SC 位写如数据。

下表列出了应用程序可能用到的系统状态变量。输入逻辑时，可以使用变量地址，也可以使用昵称。关于故障描述和如何清除故障的详细信息，参考第十二章

%S 变量

变量地址	*名称*	*定义*
%S0001	#FST_SCN	当前的扫描周期是 LD 执行的第一个周期。在停止/运行转换后第一个周期，此变量置位，第一个扫描周期完成后，结点复位。
%S0002	#LST_SCN	在 CPU 转换到运行模式时设置，在 CPU 执行最后一次扫描时清除。CPU 将这一位置 0 后，再运行个扫描周期，之后进入停止或故障停止模式。如果最后的扫描次数设为 0。CPU 停止后将%S0002 置 0，从程序中看不到%S0002 已被清 0。
%S0003	#T_10MS	0.01 秒定时结点
%S0004	#T_100MS	0.1 秒定时结点
%S0005	#T_SEC	1.0 秒定时结点
%S0006	#T_MIN	1.0 分钟定时结点
%S0007	#ALW_ON	总为 ON.
%S0008	#ALW_OFF	总为 OFF.
%S0009	#SY_FULL	CPU 故障表满了之后置 1(故障表缺省值为纪录 16 个故障，可配置)，某一故障清除或故障表被清除后，此为置 0
%S0010	#IO_FULL	I/O 故障表满了之后置 1(故障表缺省值为纪录 32 个故障，可配置)，某一故障清除或故障表被清除后，此为置 0
%S0011	#OVR_PRE	%I, %Q, %M, %G 或者布尔型的符号变量存储器发生覆盖时置 1
%S0012	#FRC_PRE	Genius 点被强制时置 1
%S0013	#PRG_CHK	后台程序检查激活时置 1
%S0014	#PLC_BAT	电池状态发生改变时，这个结点会被更新

注意: #FST_EXE 不再使用%S 地址，只用名称为“#FST_EXE”的地址标识。在停止-运行转换时，这一位置 1，表明程序块第一次被调用。

%SA, %SB 和%SC 变量

注意: 故障之后或者清除故障表之后的第一次输入扫描时，才会置位或复位%SA, %SB 和%SC 结点。也可以通过用户逻辑或使用 CPU 监控设备置位或复位%SA, %SB 和%SC 结点

变量	*名称*	*定义*
%SA0001	#PB_SUM	应用程序检测和变量检测不匹配时，这一位置位。如果故障是瞬时错误，再次向 CPU 存储程序时将这个错误清除。如果是严重的 RAM 故障，必须更换 CPU。 要清除这一位，清除 CPU 故障表或将 CPU 重新上电。
%SA0002	#OV_SWP	CPU 检测到上一个周期的扫描时间超过用户设定的时间时置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。CPU 设为固定扫描时间(Constant Sweep mode)时起作用。
%SA0003	#APL_FLT	应用程序发生故障时置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SA0009	#CFG_MM	故障表记录有配置不等故障时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SA0008	#OVR_TMP	CPU 操作温度超过正常温度(58ºC)时，这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SA0010	#HRD_CPU	自诊断检测到 CPU 硬件故障时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SA0011	#LOW_BAT	发生电池电压过低故障时置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SA0012	#LOS_RCK	扩展机架与 CPU 停止通讯时，这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SA0013	#LOS_IOC	总线控制器停止与 CPU 通讯时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0。
%SA0014	#LOS_IOM	I/O 模块停止与 CPU 通讯时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0
%SA0015	#LOS_SIO	可选模块停止与 CPU 通讯时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0
%SA0017	#ADD_RCK	系统增加扩展机架时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0
%SA0018	#ADD_IOC	系统增加总线控制器时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0
%SA0019	#ADD_IOM	机架上增加 I/O 模块时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0
%SA0020	#ADD_SIO	机架上增加智能可选模块时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0
%SA0022	#IOC_FLT	总线控制器报告总线故障，全局存储器故障或者 IOC 硬件故障时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0
%SA0023	#IOM_FLT	I/O 模块报告回路故障或者模块故障时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0
%SA0027	#HRD_SIO	检测到可选模块硬件故障时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0

变量	名称	定义
%SA0029	#SFT_IOC	I/O 控制器发生软件故障时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0
%SA0031	#SFT_SIO	.可选模块检测到内部软件错误时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0
%SA0032	#SBUS_ER	VME 总线背板发生总线错误时这一位置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0
%SA0081 – %SA0112		CPU 故障表记录了用户自定义故障时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。更多的信息见第九章服务请求 21
%SB0001	#WIND_ER	固定扫描时间模式下，如果没有足够的时间启动编程器窗口，这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SB0009	#NO_PROG	存储器保存的情况下，CPU 上电，如果没有用户程序，这一位置位。清除 CPU 故障表或者在有程序的情况下将 CPU 重新上电后，这一位清 0。
%SB0010	#BAD_RAM	CPU 上电时检测到 RAM 存储器崩溃的情况下这一位置位。清除 CPU 故障表或者在检测到 RAM 存储器正常的情况下将 CPU 重新上电后，这一位清 0。
%SB0011	#BAD_PWD	密码访问侵权时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SB0012	#NUL_CFG	试图在没有配系数据的情况下，令 CPU 进入运行模式，则这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SB0013	#SFT_CPU	检测到 CPU 操作系统软件故障时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SB0014	#STOR_ER	编程器存储操作发生故障时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SB0016	#MAX_IOC	系统配置的 IOC 超过 32 个时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SB0017	#SBUS_FL	CPU 无法访问总线时这一位置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SC0009	#ANY_FLT	有任何故障登入 CPU 或 I/O 故障表时，这一位都会置位。清除 CPU 故障表和 I/O 故障表或者将 CPU 重新上电后，这一位清 0。
%SC0010	#SY_FLT	有任何故障登入 CPU 故障表时，这一位都会置位。清除 CPU 故障表或者将 CPU 重新上电后，这一位清 0。
%SC0011	#IO_FLT	有任何故障登入 I/O 故障表时，这一位都会置位。清除 I/O 故障表或者将 CPU 重新上电后，这一位清 0。
%SC0012	#SY_PRES	只要 CPU 故障表中有故障，这一位就会置位。清除 CPU 故障表后，这一位清 0。
%SC0013	#IO_PRES	只要 I/O 故障表中有故障，这一位就会置位。清除 I/O 故障表后，这一位清 0。
%SC0014	#HRD_FLT	发生硬件故障时这一位置位。清除 CPU 故障表和 I/O 故障表或者将 CPU 重新上电后，这一位清 0。
%SC0015	#SFT_FLT	发生软件故障时这一位置位。清除 CPU 故障表和 I/O 故障表或者将 CPU 重新上电后，这一位清 0。

故障变量

故障变量在本手册第十二章讨论，但为了方便您的使用，在此列出。

系统故障变量

系统故障变量	*描述*
#ANY_FLT	重新上电或者清除 CPU/IO 故障表后的任何新故障
#SY_FLT	重新上电或者清除 CPU 故障表后的任何新的系统故障
#IO_FLT	重新上电或者清除 I/O 故障表后的任何新的 I/O 故障
#SY_PRES	说明 CPU 故障表中至少有一个故障
#IO_PRES	说明 I/O 故障表中至少有一个故障
#HRD_FLT	任何硬件故障
#SFT_FLT	任何软件故障

可配置的故障变量

可配置的故障 ***(缺省动作)***	*描述*
#SBUS_ER (诊断的)	系统总线错误 (BSERR 信号在 VME 系统总线上生成)
#SFT_IOC (诊断的)*	Genius 总线控制器不可恢复的软件故障
#LOS_RCK (诊断的)	失去机架(BRM 故障, 掉电)或错过已配置的机架
#LOS_IOC (诊断的)*	失去总线控制器或错过已配置的总线控制器.
#LOS_IOM (诊断的)	失去 I/O 模块(没有响应)或错过已配置的 I/O 模块
#LOS_SIO (诊断的)	失去智能选择模块(没有响应)或错过已配置的智能选择模块
#IOC_FLT (诊断的)	非致命总线或总线控制器错误，10 秒内超过 10 个总线错误(错误速率可配置)
#CFG_MM (致命的)	上电过程中，存储配置过程中或者运行模式时检测到模块类型错误。CPU 没有检测到独立模块(如 Genius I/O 模块)的配置参数

Non-Configurable Faults 不可配置的故障

不可配置的故障 (动作)	*描述*
#SBUS_FL (致命的)	系统总线故障。CPU 无法访问 VME 总线。BUSGRT-NMI 错误
#HRD_CPU (致命的)	CPU 硬件故障，如存储设备故障或者串口故障
#HRD_SIO (诊断的)	系统内任何硬件的非致命故障
#SFT_SIO (诊断的)	LAN 接口模块的不可恢复的软件故障
#PB_SUM (fatal)	上电过程中或者运行模式时的程序或者块检测故障
#LOW_BAT (诊断的)	CPU 或系统内其他模块的低电压信号
#OV_SWP (诊断的)	固定扫描时间超时
#SY_FULL, IO_FULL (诊断的)	CPU 故障表满了 I/O 故障表满了
#IOM_FLT (诊断的)	模块局部故障（I/O 模块的点或通道）
#APL_FLT (诊断的)	应用程序故障
#ADD_RCK (诊断的)	增加新机架，外部的或故障机架重新接入
#ADD_IOC (诊断的)	外部 I/O 总线控制器或 I/O 总线控制器重启
#ADD_IOM (诊断的)	之前的故障 I/O 模块不再有故障，或者外部 I/O 模块。
#ADD_SIO (诊断的)	增加或重启新的智能选择模块
#NO_PROG (信息)	上电时没有应用程序。只发生在 CPU 第一次上电时和电池为后备电源的 RAM 当中的程序失败
#BAD_RAM (致命的)	上电时程序存储器崩溃。程序不能被读取和/或没有通过检测
#WIND_ER (信息)	窗口不能完成错误。编程器或逻辑窗口服务跳过。固定扫描模式时发生。
#BAD_PWD (信息)	改变授权等级的请求被拒绝，密码错误。
#NUL_CFG (致命的)	转换为运行模式时，没有配置。运行时没有配置相当于延缓 I/O 扫描
#SFT_CPU (致命的)	CPU 软件故障。CPU 检测到不可恢复的错误。可能是到达看门狗定时器时间
#MAX_IOC (致命的)	超过了总线控制器允许的最大数目。CPU 最多支持 32 个总线控制器。
#STOR_ER (致命的)	从编程器向 CPU 下装数据出错，CPU 的数据可能崩溃。

程序函数如何处理数值数据

不管数据存储在位存储器还是存储在字存储器中，应用程序都可以按类型处理。

数据类型

类型	名称	描述	数据格式
BOOL	布尔	存储器的最小单位。由两种状态，1 或者 0。布尔数列长度为 N.	
BYTE	字节	8 位二进制数据。范围 0~255。字节数列长度为 N	
WORD	字	16 个连续数据位。字的值的范围是 16 进制的 0000~FFFF.	寄存器 1 (16 位状态) 16 1
DWORD	双字	除了不用 16 位连续数据位，而用 32 位连续数据位，与单字类型书具有同样的特性	寄存器 2 寄存器 1 32 17 16 1 (32 bit 状态)
UINT	无符号整型	占用 16 位存储器位置。正确范围 0~65535(16 进制 FFFF)	寄存器 1 (二进制值) 16 1
INT	带符号整形	占用 16 位存储器位置。补码表示法。带符号整型数正确范围为–32768~ +32767	寄存器1 (补码值) S 16 1 s=符号位 (0=正, 1=负)
DINT	双精度整型	占用 32 位存储器位置 (两个连续的 16 位存储器位置)。用最高位表示数值的正负。带符号双整型数 (DINT)正确范围为-2147483648~ +2147483647	Register 2 Register 1 S 32 17 16 1 (Binary value) s=符号位 (0=正, 1=负)
REAL	浮点	占用 32 位存储器位置 (两个连续的 16 位存储器位置)。这种格式存储的数据范围为± 1.401298E-45 to ±3.402823E+38。关于 IEEE 格式，参考“实型数”	寄存器 2 寄存器 1 32 17 16 1 (IEEE 格式)
BCD-4	4 位 BCD	占用 16 位存储器位置。用 4 个二进制数表示 0~9 的一个十进制数(BCD)。4 位的 BCD 码表示范围为 0~9999	寄存器1 4 3 2 1 (4 BCD 位) 13 9 5 1
BCD-8	8 位 BCD	占用 2 个连续的 16 位存储器位置(32 个连续位)。用 4 个二进制数表示 0~9 的一个十进制数(BCD)。8 位的 BCD 码表示范围为 0~99999999	Register 2 Register 1 8 7 6 5 4 3 8 1 32 29 25 21 17 16 13 9 5 1 (8 BCD digits)

类型	名称	描述	数据格式
MIXED	混合型	只用于乘/除函数。乘函数由两个整型输入，一个双整型结果。除函数有一个输入量为双整型被除数，另一个输入量为整型除数，输出的结果为整型量。	16 16 = 32 32 16 = 16
ASCII	ASCII	用 8 位二进制数表示一个字符。一个字由两个(打包的)ASCII 字符表示。第一个字符对应低字节。每一部分的剩余七位被转换。	

注意: 使用没有指明位类型的函数会影响被写入的字节/字/双字的位的转换。关于如何使用浮点数，参考 7-17 页"实型数"

实型数

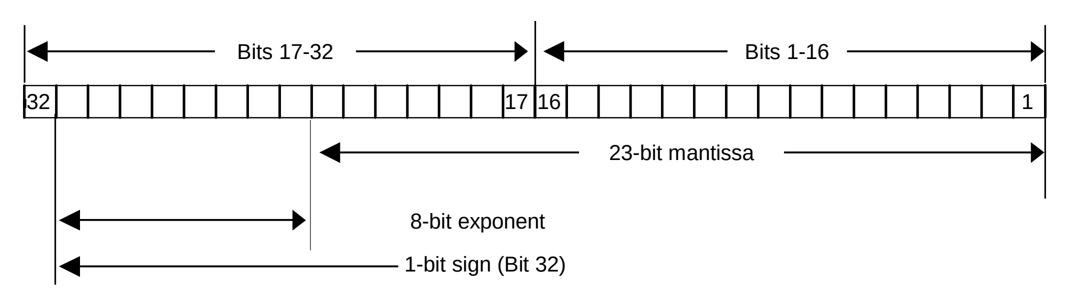
实型数可以存储 32 位小数，实际上就是浮点数。浮点数以单精度的 IEEE 标准格式存储。这种格式使用两个相邻的 16 位字。

实型数典型应用:存储模拟量 I/O 设备数据，计算结果和常数

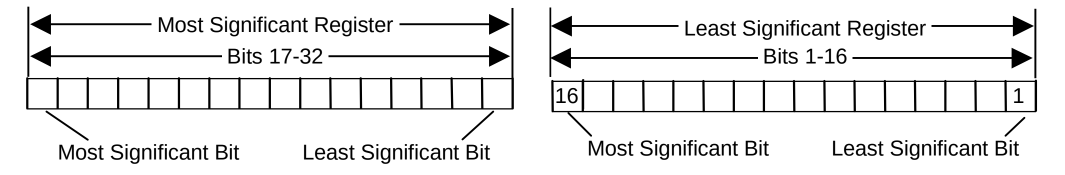
.实型数精度为 6~7 个有效位，精度范围大概是$\pm 1.401298\text{x}10^{-45}$ ~ $\pm 3.402823\text{x}10^{38}$

注意: 编程软件允许将 PAC 系统项目的 32 位值(DWORD, DINT, REAL)以离散存储器%I,%M 和%R 存储器存储。90-70 系列项目不允许。(注意，位变量地址存储的非位参数必须是按字节排列的，这和 90-70 系列 CPU 相同)

实型数的内部格式



浮点数如下表使用。在这个表中，如果浮点数使用了 R5 和 R6，例如，R5 时低位字 R6 时高位字。



浮点数与操作的错误

当数值大于 3.402823E+38 或小于–3.402823E+38 时会发生溢出。发生溢出时，函数的 enable out 输出被设为 Off；结果大于 3.402823E+38 时被设为正无穷大，结果小

于–3.402823E+38 时被设为负无穷大。可以测试 enable out 输出来确定是否发生溢出。

POS_INF = 7F800000h – IEEE 正无穷(16 进制)

NEG_INF = FF800000h – IEEE 负无穷(16 进制)

如果溢出产生的无穷大用作其他实型函数的操作数，会产生未定义的结果。这个未定义结果指的是不是一个数(NaN)，当 ADD_REAL 函数将正无穷或负无穷作为操作数时，会产生 NaN。

NaN 值:

7F800001~7FFFFFFF

FF800001~ FFFFFFFF

函数的任何一个操作数是 NaN 时，结果一般为 NaN.

注意: 对于 NaN ，Enable Out 输出为 Off(不得电)

逐字替换

对程序的某些修改不会改变整个程序的大小，这种修改称为逐字替换。例如更改结点和线圈的类型或者更改已经存在的函数块的某个变量的地址

符号变量

创建，删除或修改符号变量定义不属于逐字替换。

下列情况属于逐字替换：

- 在两个符号变量之间转换
- 在符号变量和映射变量之间转换
- 在符号变量和常数之间转换

第8章 指令系统

本章描述的是在 PACSystems 控制系统中用来创建梯形图逻辑程序的程序设计指令。

P8-2 提供了和指令一起使用的操作数类型总览。

PACSystems 梯形图逻辑指令系统包括下列类型的指令：

指令操作数

PACSystems 指令操作数和功能有下列形式：:

- 常量
- 位于 PACSystems 存储区域的变量(%I, %Q, %M, %T, %G, %S, %SA, %SB, %SC, %R, %W, %L, %P, %AI, %AQ)。
- 符号变量
- 参数化块或 C 块的参量
- 能流
- 数据流
- 计算基准，如间接基准或字位基准

操作数的类型和长度必须与它进入的参数的类型和长度相一致。另外，对操作数尽可能少的限制，早期的 GE Fanuc PLC 上的限制已经在 PACSystems PLC 上解除。PACSystems 指令和功能有下列操作数约束：

- 常量不能用作输出参数的操作数，因为输出的值不能写入常量。
- 位于%S 存储器内的变量不能用作输出参数的操作数，因为%S 存储器是只读类型的。
- 位于%S, %SA, %SB, and %SC 存储器中的变量不能用作数字表示的参数的操作数，例如 INT（单精度符号整数）, DINT（双精度符号整数）, REAL（浮点数），等。
- 当一些功能模块在计算过程中，写一个值到它的输入参数时，数据流禁止用于这些功能模块的输入参数中，数据流在这种情况下被禁止因为，数据流存储在临时存储器中，任何赋给它的更新值都不允许进入用户程序。
- EN, OK 的参数和许多其他的布尔型输入输出参数在能流中受限制。
- 使用带参数块或者外部模块参数作为指令或功能的操作数的一些限制将在第六章描述。
- 离散型存储器变量(I, Q, M,和 T)必须以字节形式排列。

注意下列事项:

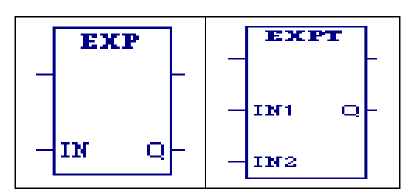
- 可用于所有定向单字的存储器(%R, %W, %P, %L, %AI, %AQ)的间接基准可以作为指令自变量使用，允许把变量放在任何相应的定向单字存储器里。注意，间接基准在进入一个指令或功能前，立即转化为相应的直接基准。
- 一般情况下，字位基准允许用于触点或线圈指令（不包括跳变触点和线圈）。也被允许作为接受单个或没有排列位的功能参数的自变量。

高等数学函数

高等数学函数执行对数，指数，平方根，三角函数和反三角函数的操作。

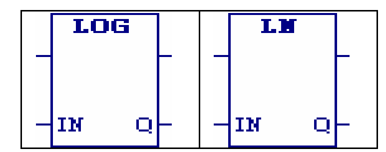
函数	助记符	描述
指数	EXP	计算 e^{IN}，IN 为操作数。
	EXPT	计算 $IN1^{IN2}$。
反三角函数	ACOS	计算 IN 操作数的反余弦，以弧度形式表达结果。
	ASIN	计算 IN 操作数的反正弦，以弧度形式表达结果。
	ATAN	计算 IN 操作数的反正切，以弧度形式表达结果。
对数	LN	计算 IN 操作数的自然对数。
	LOG	计算 IN 操作数的 10 为底的对数。
平方根	SQRT_DINT	计算操作数 IN 的平方根，一个双精度整数。结果的双精度整数部分存到 Q 中。
	SQRT_INT	计算操作数 IN 的平方根，一个单精度整数。结果的单精度整数部分存到 Q 中。
	SQRT_REAL	计算操作数 IN 的平方根，一个实数。实数结果存到 Q 中。
三角函数	COS	计算操作数 IN 的余弦，IN 以弧度表示。
	SIN	计算操作数 IN 的正弦，IN 以弧度表示。
	TAN	计算操作数 IN 的正切，IN 以弧度表示。

指数/对数函数



当一个指数/对数功能块使能激活，它对输入的实数值执行适当的操作，把结果存入输出 Q 中。

- 对以 e 为底的指数函数(EXP)，计算 e 的 IN 次方，结果存入 Q。
- 对 X 次幂函数(EXPT)，计算 IN1 的 IN2 次方，结果存入 Q。
- 对以十为底的对数函数(LOG)，计算 10 为底的 IN 的对数，结果存入 Q。
- 对自然对数函数(LN)，计算 e 为底的 IN 的对数，结果存入 Q。



在没有溢出的情况下，这些功能块能流输出激活，除非有下列情况之一发生：]

- 在 LOG 或 LN 中，IN < 0。
- 在 EXPT 中，IN < 0。
- 在 EXP 中，IN 是负无穷大。
- IN，IN1，IN2 不是数字。

指数/对数函数的操作数

参数	描述	许用操作数	可选性
IN 或 IN1	在 EXP, LOG, LN 中，IN 是一个实数值。 EXPT 功能块有两个输入 IN1 和 IN2，IN1 为底，IN2 为指数。	除位于%S—%SC 里的变量外的所有操作数	No
IN2 (EXPT)	在 EXPT 中的指数。	除位于%S—%SC 里的变量外的所有操作数	No
Q	为 IN，IN1 和 IN2 计算出的指数/对数的值。	除常数和位于%S—%SC 里的变量外的所有操作数	No

平方根函数



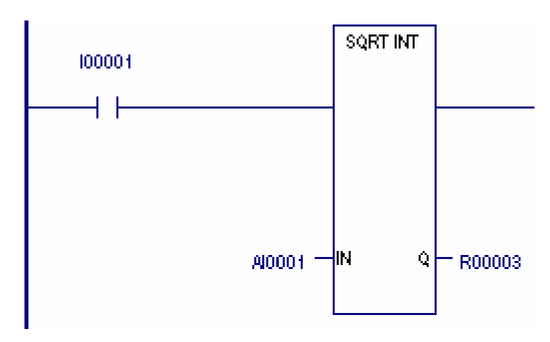
当平方根功能块使能激活，它计算出 IN 的平方根并把结果存到 Q 中。Q 必须和 IN 是同一种数据类型。

在没有溢出的情况下，这些功能块能流输出激活，除非有下列情况之一发生：

- IN < 0
- IN 不是数字。

范例

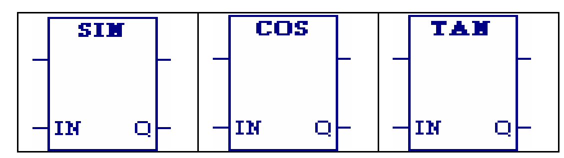
只要%I00001 在 ON 状态，计算存储单元%AI0001 里的整数的平方根并把结果送进存储单元%R00003 里。



平方根函数的操作数

参数	描述	许用操作数	可选性
IN	平方根计算的输入值，如果 IN < 0，功能块不能传递能流。	除位于%S—%SC 里的变量外的所有操作数	No
Q	计算出的平方根。	除常数和位于%S—%SC 里的变量外的所有操作数	No

三角函数



SIN, COS 和 TAN 功能块用来计算输入为弧度的正弦，余弦和正切值。当这些功能模块接收到能量流，它计算 IN 的正弦值(余弦或正切值)并把结果存入输出 Q。

SIN, COS 和 TAN 模块能接受范围很大的输入值，$-2^{63} < IN < +2^{63}$, ($2^{63} \approx 9.22x10^{18}$)

功能块能流输出激活，除非有下列情况之一发生：

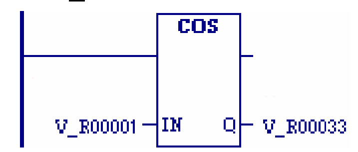
- 在 TAN 中，IN = π/2, 3*π/2, 5*π/2 等等。
- IN 不是一个数字。

操作数

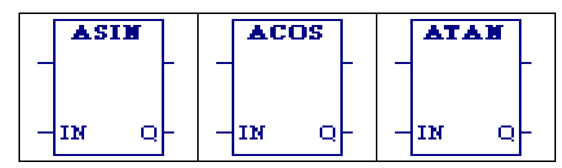
参数	描述	许用操作数	可选性
IN	弧度为单位的数字，$-9.22x10^{18} < N < 9.22x10^{18}$ ($9.22x10^{18} \simeq 2^{63}$.) (实数)	除位于%S—%SC 里的变量外的所有操作数	No
Q	IN 的三角函数值(实数)	除常数和位于%S—%SC 里的变量外的所有操作数	N

范例

在 V_R00001 中的值的余弦被放在 V_R00033 中



反三角函数 – ASIN, ACOS, 和 ATAN



当一个反正弦 (ASIN), 反余弦(ACOS), 或反正切(ATAN)函数接收到 power flow 时,分别地计算 IN 的反正弦、反余弦或反正切，把计算结果以弧度为单位存储在输出点 Q 中。IN 和 Q 都是实数。ASIN 函数和 ACOS 函数的输入范围小，－1＜IN＜1。给 IN 参量一个有效值，ASIN_REAL 函数产生一个如下的计算结果 Q：

$$ASIN(IN) = -\frac{\pi}{2} \leq Q \leq \frac{\pi}{2}$$

ACOS_REAL 函数产生一个如下的计算结果 Q:

$$ACOS(IN) = -0 \leq Q \leq \pi$$

ATAN 函数的输入范围最大，$-\infty \leq IN \leq +\infty$。给 IN 参量一个有效值，ATAN_REAL 函数产生一个如下的计算结果 Q：

$$ATAN(IN) = -\frac{\pi}{2} \leq Q \leq \frac{\pi}{2}$$

功能块能流输出激活，除非有下列情况之一发生：

- 在做 ASIN,ACOS,或 ATAN 运算时， IN 在有效范围之外。
- IN 不是一个数字。

反三角函数的操作数

参量	*描述*	*许用操作数*	*可选性*
IN	实数 ASIN 和 ACOS: -1 ≤ IN ≤ 1 ATAN: $-\infty \leq IN \leq +\infty$	除位于%S—%SC 里的变量外的所有操作数	No
Q	IN 的三角函数值（实数）。ASIN: (-π/2) ≤ Q ≤ (π/2) ACOS: 0 ≤ Q ≤ π ATAN: (-π/2) ≤ Q ≤ (π/2)	除常数和位于%S—%SC 里的变量外的所有操作数	No

位操作功能

LD 位操作功能对位串执行比较、逻辑运算和传送操作。

功能	*助记符*	*描述*
位位置	BIT_POS_DWORD BIT_POS_WORD	位位置. 在位串里找出一个被置 1 的位。
位排序	BIT_SEQ	位排序. 排好一个位串值, 起始于 ST. 通过一个位数组操作一个位序移位。容许最大长度 256 字。
位置位,清除	BIT_SET_DWORD BIT_SET_WORD	位置位. 把位串中一个位置 1。
	BIT_CLR_DWORD BIT_CLR_WORD	位清除. 通过把位串里一个位置 0 清除该位。
位测试	BIT_TEST_DWORD BIT_TEST_WORD	位测试. 测试位串里的一个位，测定该位当前是 1 或 0。
逻辑“与”	AND_DWORD AND_WORD	逐位比较位串 IN1 和 IN2。当相应的一对位都是 1 时，在输出位串 Q 相应位置放入 1，否则，在输出位串 Q 相应位里放 0。
逻辑取反	NOT_DWORD NOT_WORD	逻辑取反. 把输出位串 Q 每个位的状态置成与位串 IN1 每个相对应位相反的状态。
逻辑“或”	OR_DWORD OR_WORD	逐位比较位串 IN1 和 IN2。当相应的一对位都是 0 时，在输出位串 Q 相应位置放入 0，否则，在输出位串 Q 相应位里放 1。
逻辑“异或”	XOR_DWORD XOR_WORD	逐位比较位串 IN1 和 IN2，当相应的一对位不同时，在输出位串 Q 相应位置放入 1，当相应的一对位相同时，在输出位串 Q 相应位里放 0。
屏蔽比较	MASK_COMP_DWORD MASK_COMP_WORD	屏蔽比较。用屏蔽选择位的能力比较两个单独的位串。
位循环	ROL_DWORD ROL_WORD	左循环.一个固定位数的位串里的位循环左移。
	ROR_DWORD ROR_WORD	右循环.一个固定位数的位串里的位循环右移。
位移位	SHIFTL_DWORD SHIFTL_WORD	左移位. 一个固定位数的字或字串里的位左移。
	SHIFTR_DWORD SHIFTR_WORD	右移位. 一个固定位数的字或字串里的位右移。

位操作功能的数据长度

位操作功能对 1 到 256 个占用相邻内存位置的 WORD 或 DWORD 数据执行操作。

位操作功能把 WORD 或 DWORD 数据当作一个连续的位串，第一个 WORD 或 DWORD 的第一位是最低位（LSB），最后一个 WORD 或 DWORD 的最后一位是最高位（MSB）。例如，如果从%R0100 开始指定 3 个 WORD 数据，那么这 3 个 WORD 数据被当作 48 个连续位。

%R0100	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	← bit 1 (LSB)
%R0101	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	
%R0102	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	
	↑ (MSB)																

警告

在对多字数据执行操作时，不推荐使用重叠输入和输出基准地址，因为这样会产生意外。

注意：所有的功能块(Bit Test, Bit Set, Bit Clear, and Bit Position) 回送一个位位置指示作为输出参量，正如在上面表中所显示的，位位置编号方式从 1 开始，而不是从 0 开始。

位位置功能

BIT POS DWORD ?? IN Q POS

BIT POS WORD ?? IN Q POS

位位置功能在一个位串里查找一个置 1 位的位置。

每一次扫描到功能块使能激活，位位置功能从 IN 开始扫描位串。当停止扫描时，不是等于 1 的位被发现，就是整个位串被扫描结束。POS 被置为在位串里第一个非零位的位置数，如果位串里没有非零位，POS 置 0。

串长可以在 1 到 256 个 WORD 或 DWORD 之间选择。只要位位置功能激活，它就传递向右传递能流。

操作数

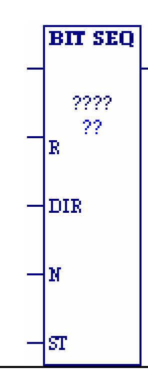
参量	*描述*	*许用操作数*	*可选性*
长度（Length）(??)	在位串里 WORD 或 DWORD 的数量. 1 ≤ Length ≤ 256.	常数	No
IN	执行操作的数据	所有的操作数. 常数仅可以用在长度为 1 的情况。	No
Q	如果一个被置 1 的位被发现，激活。	流	Yes
POS	一个无符号的整数。给出位串里第一个非零位的位置，如果位串里没有非零位，则置 0。	除常数和位于%S—%SC 里的变量外的所有操作数。	No

范例

当 V_I00001 在“开”状态, 搜索起始于 V_M00001 的位串，当有一个等于 1 的位被发现，或者 6 个字被搜索完，则停止搜索。线圈 V_Q00001 接通。如果一个等于 1 的位被搜索到，它在位串里的位置写进 V_AQ0001，同时 V_Q00002 接通。例如，如果 V_I00001 在“开”状态，位 V_M00001 是 0，V_M00002 是 1，那么，写进 V_AQ0001 的值是 2。

V_I00001 BIT POS WORD 6 V_Q00001 V_Q00002 V_M00001 IN Q POS V_AQ0001

位排序



位排序(BIT_SEQ)功能执行一连串的连续位从头到尾位序变化功能。

BIT_SEQ 的操作取决于复位输入（R）的值、使能输入（EN）的当前值和前次值。

R 当前执行指令	*EN 上次执行指令*	*EN 当前执行指令*	*位序器执行指令*
ON	ON/OFF	ON/OFF	位排序功能块复位
OFF	OFF	ON	位排序功能块作增 1/减 1 操作
		OFF	位排序功能块不执行操作

当复位输入 (R)激活，则使能输入 (EN) 无效，复位位排序功能块. 这时，, 设定当前步数是可选操作数 N 的值。如果无指定操作数 N，步数置为 1。在位排序功能块里，除了当前步数指向的位被置为 1，其余所有的位及 ST 都被置为 0。

当使能输入（EN）激活而 R 不激活，并且 EN 前次状态是 OFF，当前步数指向的位清零。当前步数根据 DIR 操作数增加或减少。然后新步数指向的位被置为 1。

- 当步数增加到超出范围(1 ≤ 步数 ≤ Length)，步数被置回 1。
- 当步数减少到超出范围(1 ≤ 步数 ≤ Length)，步数被置为 Length 值。

参量 ST 是可选的。如果不用它，除非没有位被置 1 或清零，否则位排序功能块按上述操作。位排序功能块当前步数只能在允许的范围内循环。

当位排序功能块使能激活时，把能流传递到右边。.

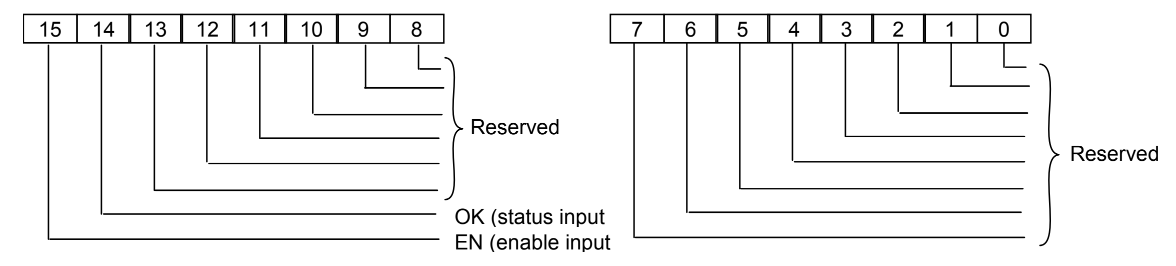
位排序功能块必需的内存

每个位排序功能块使用控制块内的一个 3 个字数组。控制块可以是一个带符号的变量，也可以是在%R, %W, %L, 或 %P 存储器内的变量。

字 1	当前步数
字 2	位列长度
字 3	控制字

注意: 不要从其他功能块写入控制块存储寄存器。

字 3 (控制字)以下面的格式存储与功能块有关的 BOOL 量输入和输出的状态。



注意:

- 第 0 位到 13 位不用。.
- 操作数 N 的输入从 1 到 16，而不是从 0 到 15。

位排序功能的操作数

Warning

不要用其他指令写入控制块存储器，重叠给定引起 **BIT_SEQ** 的不确定的操作。

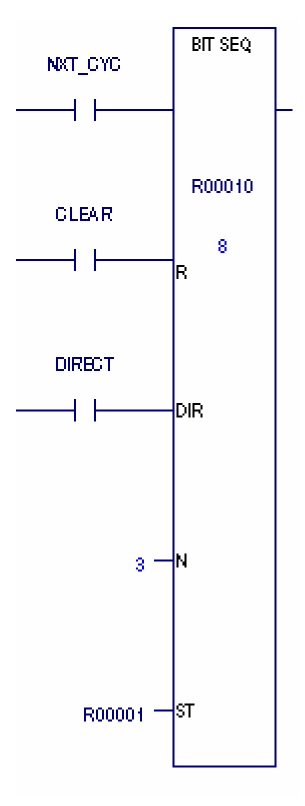
参量	*描述*	*许用操作数*	*可选性*
地址 (????)	控制块的起始地址，是一个 3 个字的数组。: 字 1: 当前步数 字 2: 位列长度 字 3: 控制字, 跟踪使能 的最后状态和到右边的能流 的状态。	带符号变量，在存储器%R, %W, %P, or %L 中的变量。	No
长度 (??)	在位排序功能块里的位数，ST，也是 BIT_SEQ 的移步范围。1 ≤ Length ≤ 256.	常数	No
R	当 R 激活时，把 N 里的值置入 BIT_SEQ 的步数（缺省值为 1），在位排序功能块里，除了当前步数指向的位被置为 1，其余所有的 ST 位都被置为 0。	流	No
DIR	当 DIR 激活时，BIT_SEQ 的步数优先移位（shift ）做增 1 操作，否则，做减 1 操作。	流	No
N	当 R 激活时，置入 BIT_SEQ 的步数的值。缺省值为 1. 1 ≤ N ≤ Length. 如果 N < 1, R 激活时步数回 1. 如果 N > Length,步数回 Length.	除了位于%S-%SC 中的变量之外的所有操作数。	Yes

参量	*描述*	*许用操作数*	*可选性*
ST	包含排序功能的第一个字。如果 ST 不用，除了没有位被置 1 或清零外，BIT_SEQ 做前述操作，BIT_SEQ 当前步数（在控制块字 1 里）只能在允许的范围内循环。 如果 ST 在%M 存储器里，长度是 3 的话，BIT_SEQ 占用 3 位，字节里的其他 5 个位不用。如果 ST 在%R 存储器里，长度是 17 的话，BIT_SEQ 用完%R1 和 %R2 存储器全部的 4 个字节。	除了常数、流（flow）和位于%S 中的变量之外的所有操作数	Yes

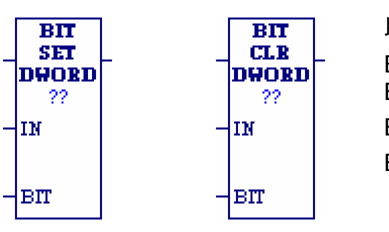
范例

BIT_SEQ 对存储寄存器 %R00001 的操作。它的静态数据存储在存储器 %R0011 和 %R0012 中。当 CLEAR 激活，BIT_SEQ 复位，当前步数置位 N 中指定的 3，%R0001 的第 3 位置为 1，其余位置为 0。

当 NXT_CYC 激活而 CLEAR 不激活，步数 3 的位清零，步数 2 或步数 4 的位置 1（取决于 DIR 是否激活）。



位置位，清零



助记符

BIT_SET_DWORD
BIT_SET_WORD
BIT_CLR_DWORD
BIT_CLR_WORD

位置位 (BIT_SET_DWORD 和 BIT_SET_WORD)功能是把位串中的一个位置 1。位清零功能是通过把位串中一个位置 0 来清除该位

每次使能激活，该功能置位或清零指定位。当一个变量是大于用来指定位数的常数，通过连续扫描可以对不同的位置位或清零。一旦一个位被置位或清零，这个位的状态被刷新，而位串里其他位的状态不受影响。

该功能传递能量流到右边，除非位的值在规定的范围之外。

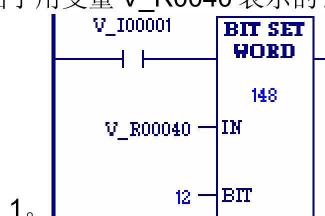
操作数

参量	*描述*	*许用操作数*	*可选性*
长度（Length）(??)	在位串里 WORD 或 DWORD 的数目 1 ≤ Length ≤ 256.	常量	
IN	要处理的数据第一个 WORD 或 DWORD	除了常数、流（flow）和位于%S 中的变量之外的所有操作数	
BIT	在 IN 里置位或清零的位数. 对于 WORD 1 ≤ BIT ≤ (16 * Length). 对于 DWORD 1 ≤ BIT ≤ (32 * length)	除了%S-%SC 中的变量之外的所有操作数	

范例

例 1

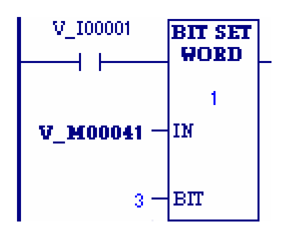
只要输入 V_I0001 被置位, 开始于用变量 V_R0040 表示的%R00040 的位串的第 12 位被置 1。



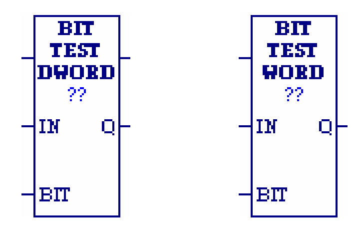
例 2

只要 V_I00001 被置位，开始于%M00041 的位串的第 3 位%M00043 被置 1。

注意：任何其他的和%M00043 字节相同的存储器单元（象%M00041, %M00042, %M00044 等），如果既没有状态值也没有转换值，也受 BIT_SET 功能影响。



位测试



当位测试功能块有使能输入，它将对位串里的一个位进行测试，确定该位当前状态是 1 或 0，并把测试结果放在输出点 Q 中。

每次扫描有使能输入，位测试功能把输出点 Q 置为和指定位相同的状态，当一个寄存器大于用来指定位数的常数，通过连续扫描可以对不同的位置进行测试。如果 BIT 的值超出范围（1 ≤ BIT ≤ (16 * length)，用于 WORD；或 1 ≤ BIT ≤ (32 * length)，用于 DWORD），那么，Q 点置为“OFF”。

可以在 1 到 256 个 WORD 或 DWORD 之间确定一个串长。

注意:　当使用位测试功能时，位的编码从 1 到 16，而不是从 0 到 15（用于 WORD），或从 1 到 32（用于 DWORD）。

操作数

参量	*描述*	*许用作数*	*可选性*
长度（Length）(??)	进行测试的数据串中的 WORD 或 DWORD 的数目，1 ≤ Length ≤ 256.	常量	No
IN	进行测试的数据串中第一个 WORD 或 DWORD。	任何操作数	No
BIT	IN 中要进行测试的位数. 1 ≤ BIT ≤ (16*Length).	除了在变量%S - %SC 中的任何操作数。、	No
Q	测试过位的状态。如果测试过的位是 1，那么 Q 有输出。	流（Flow ）	No

范例

例 1

只要输入 V_I0001 被置位，就测试由 PICKBIT 决定的位置的位，该位在 PRD_CDE 所代表的字串中。如果该位是 1，输出点 Q 有输出，激活 ADD 功能块，使 ADD 功能块输入端 IN1 的当前值加 1。

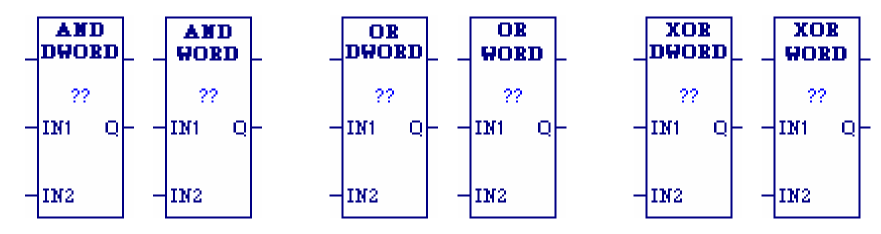
V_I00001
BIT TEST WORD
1
PRD_CDE — IN Q
PICKBIT — BIT
ADD UINT
FND_ON — IN1 Q — FND_ON
1 — IN2

例 2

只要输入 V_I0001 被置位，就测试由 PICKBIT 决定的位置的位，该位在 PRD_CDE 所代表的字串中。如果该位是 1，输出点 Q 有输出同时线圈 V_Q0001 接通。

V_I00001
BIT TEST WORD
43
V_Q00001
PRD_CDE — IN Q
PICKBIT — BIT

逻辑“与”，逻辑“或”，逻辑“异或”



每次逻辑运算功能块有使能输入，逻辑运算功能检查在 IN1 和 IN2 位串中相应的位，从位串最小有效位开始。串长可以确定在 1 到 256 个 WORD 或 DWORD 之间。IN1 和 IN2 位串可以部分重叠。

逻辑“与”

如果 AND 功能检查的两个位都是 1， AND 功能块在输出位串 Q 中相应的位置放入 1。如果这两个位有一个是 0 或者两个都是 0，AND 功能块在输出位串 Q 中相应的位置放入 0。

AND 功能块只要使能激活，就传递能流。

提示: **可以利用逻辑 AND 功能屏蔽或筛选位，仅有某些对应于屏蔽控制字中 1 的位状态信息可以通过，其他位被置 0。**

逻辑“或”

如果逻辑 OR 功能块检查的任一位是 1，逻辑 OR 功能块在输出位串 Q 中相应的位置放入 1. 如果两个位都是 0，逻辑 OR 功能块在输出位串 Q 中相应的位置放入 0。逻辑 OR 功能使能激活，就向右传递能流。。

提示:

- 可以利用逻辑 OR 功能用一个简单的逻辑结构组合串或者控制很多输出。逻辑 OR 功能和并联乘法算法中两个继电器触点作用相同。
- 可以利用逻辑或功能根据输入点状态直接驱动指示灯，或使状态灯闪烁。

逻辑“异或”

当逻辑“异或”功能块接收到信息流，就对位串 IN1 和 IN2 中每个相应的位进行比较。如果某对位的状态不同，逻辑 XOR 功能块就在输出位串 Q 中相应的位置放入 1。

对某对位做比较时，如果只有一个位是 1，逻辑“异或”功能块就在输出位串 Q 中相应的位置放入 1。逻辑 XOR 功能块使能激活，就向右传递能流。

提示

- 如果输入位串 IN2 和输出位串 Q 从相同的基准地址开始，在输入位串 IN1 中是 1 的位将导致输入位串 IN2 中相应的位在 0 和 1 之间交替变化，每次变化发生在功能块接收到能的时候。
- 通过以两倍的闪烁速率用脉冲输送能流到功能块，可以设计一个较长的周期。 能流脉冲应该是一个扫描周期长。（单触发线圈或自复位定时器）
- 可以利用逻辑 XOR 快速比较两个位串，或者使一个组位以每两次扫描一次 ON 的速率闪烁。
- 逻辑 XOR 可用于透明屏蔽。

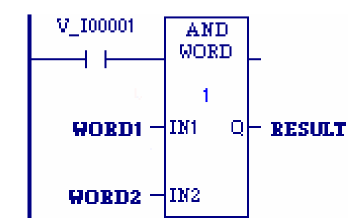
逻辑“与”、逻辑“或”、逻辑“异或”功能的操作数

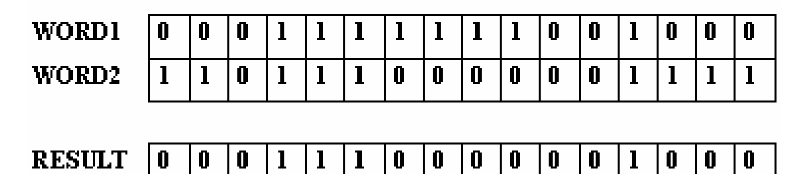
参量	*描述*	*许用操作数*	*可选性*
长度（Length ）(??)	在位串中执行逻辑操作的字数。 1 ≤ Length ≤ 256.	常量	No
IN1	被操作的第一个输入串的第一个 WORD（DWORD）	任何操作数	No
IN2 (必须和 IN1 的数据类型相同.)	被操作的第二个输入串的第一个 WORD（DWORD）	任何操作数	No
Q (必须和 IN1 的数据类型相同)	操作结果的第一个 WORD 或 DWORD。	除了常数和在存储器%S 中变量的任何操作数	No

范例

逻辑 AND

当输入变量 v_I0001 被置 1，功能块对由变量 WORD1 和 WORD2 所表示的 16 位位串进行比较检查。检查结果放在输出位串 RESULT 中。

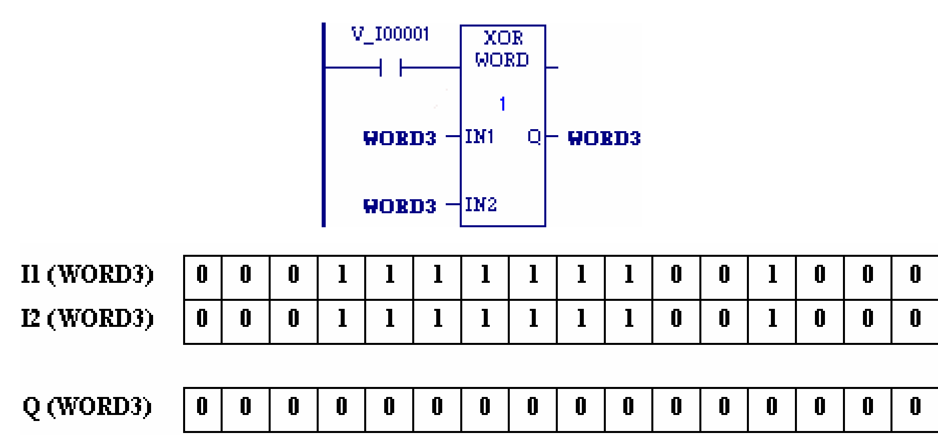




WORD1	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
WORD2	1	1	0	1	1	1	0	0	0	0	0	0	1	1	1	1
RESULT	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0

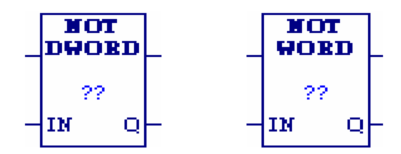
逻辑 XOR

只要输入变量 V_I0001 被置 1，由变量 WORD3 所表示的位串被清零（所有的位被置 0）。



I1 (WORD3)	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
I2 (WORD3)	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
Q (WORD3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

逻辑取反



当逻辑 NOT 功能块接收到能流，功能块就把输入位串 IN1 中每一位的状态置成其相反状态，放在输出位串 Q 中相应的位里。

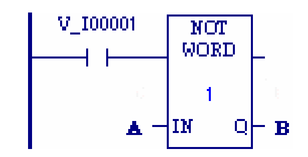
每次逻辑 NOT 功能块接收到 power，所有的位的状态都被改变，使输出位串 Q 与输入位串 IN 成为逻辑互补。只要逻辑 NOT 功能块接收到 power，就向右传递能流。串长可以被指定在 1 到 256 个 WORD 或者 DWORD 之间。

操作数

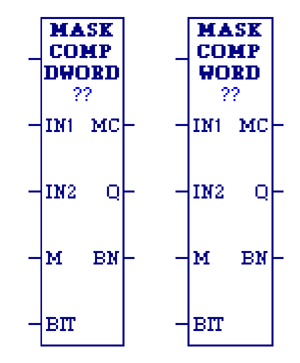
参量	描述	容许操作数	可选性
长度（Length ）(??)	输入到 NOT 功能块的位串里的 WORD 或 DWORD 的数目。1 ≤ Length ≤ 256.	常量	No
IN1	输入到 NOT 功能块的位串里的第一个 WORD 或 DWORD。	任何操作数	No
Q (必须和 IN1 的数据类型相同)	取反结果的第一个 WORD 或 DWORD。	除了常数和在存储器%S 中变量的任何操作数	No

范例

只要输入变量 V_I0001 被置 1，逻辑 NOT 功能块就对由变量 A 所表示的位串进行取反操作，结果存放在输出变量 B 中。变量 A 保持原来的值不变。



屏蔽比较



T 屏蔽比较功能模块(MASK_COMP_DWORD 和 MASK_COMP_WORD)比较两个位串的内容。它可以屏蔽某些选择的位。

提示：输入位串 1 可以包含诸如线圈或者电机启动的输出状态。输入字串 2 可以包含他们的输入反馈状态，例如限位开关或触点

当功能模块接收到能量流，它开始比较第一个位串和第二个位串的相对应的位。这种比较将持续下去，直到不可比较或者位串比较结束。

BIT 输入存储了下一个比较开始时的位编号。通常它和最后一个不可比较的位编号相同。因为最后一个不可比较的位编号存储在输出 BN 中，所以相同的基准地址可以用在 BIT 和 BN 上。一个比较通常在 BIT 后的第一位开始；因此最初的 BIT 的值应该小于开始比较的位的编号(例如 0，在%I00001 开始比较)。在 BIT 和 BN 中用相同的基准地址将导致：在不可比较发生后，比较将在接下来的位的位置开始；或者，如果所有的位在接下来的模块启动时成功比较，比较将从头开始。

提示:　如果在位串的其他位置开始下一个比较，可以输入不同的 BIT 和 BN 基准地址。如果 BIT 的值超出了位串的长度，在第二次比较前 BIT 会被置 0。

只要模块使能激活，就向右传递能流。模块的其他输出将取决于相应的屏蔽位的状态。

如果 IN1 和 IN2 中所有的相对应的位匹配，模块将设置不可比较位的输出 MC 为 0 并置 BN 为输入位串的最高位的编号。然后比较停止。在第二次屏蔽比较启动时，它被重设为 0。

如果不可比较位被搜索到，即如果两个正在比较的位不相同，那么模块将检查位串 M 中相对应的位

如果屏蔽位为 1，则比较将继续，直到搜索到另一个不可比较位或者输入位串比较结束。

如果一个不可比较位被检测到并且相应的屏蔽位为 0，模块将执行下列操作：

1. 在 M 中设置相应的屏蔽位为 1
2. 设置 MC 输出为 1
3. 更新输出位串 Q 来匹配新的位串 M
4. 设置位编号输出(BN)为不可比较位的编号
5. 停止比较

屏蔽比较功能模块的操作数

参数	*描述*	*许用的操作数*	*可选性*
Length (??)（长度）	两个比较串的 DWORD 或 WORD 数量 DWORD: 1 ≤ Length ≤ 2,048 WORD: 1 ≤ Length ≤ 4,096	常数	No
IN1	被比较的第一个字串	任何操作数。当长度为 1 时，常数合法	No
IN2	被比较的第二个字串	任何操作数。当长度为 1 时，常数合法	No
M	位串屏蔽，包含了正在进行比较的状态	除了流或在%S 存储器中变量外的任何操作数。当长度为 1 时，常数合法	No
BIT	BIT+1 等于下一次比较开始时的位数字	除了在%S- %SC 存储器中变量外的任何操作数	No
Q	屏蔽比较串的输出	除了常数外的任何操作数。	No
BN	最后一个不可比较位的编号，或者当没有不可比较位时，输入中最高位的编号	除了常量和%S 存储器中变量外的任何操作数	No
MC	如果有不可比较位时使用	流	Yes

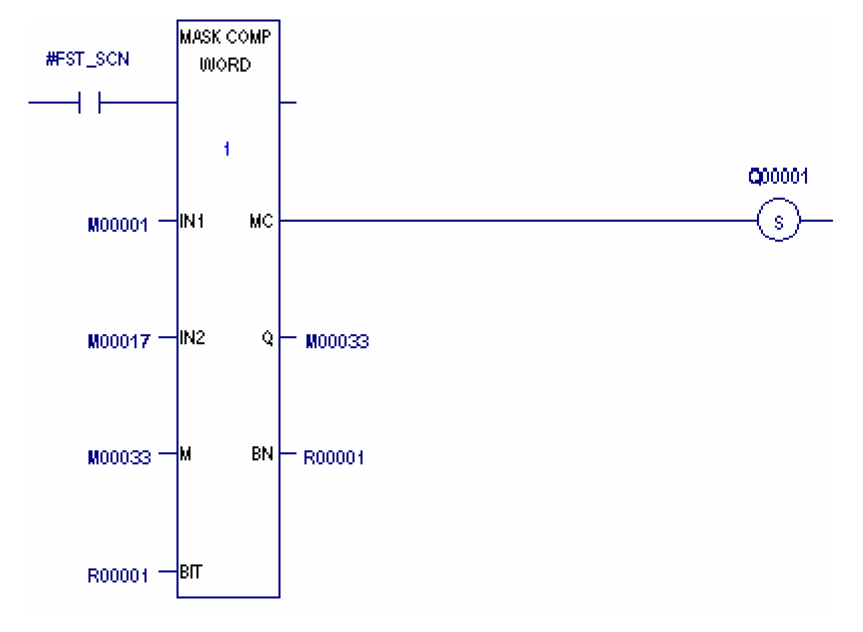
举例说明屏蔽比较

例 1

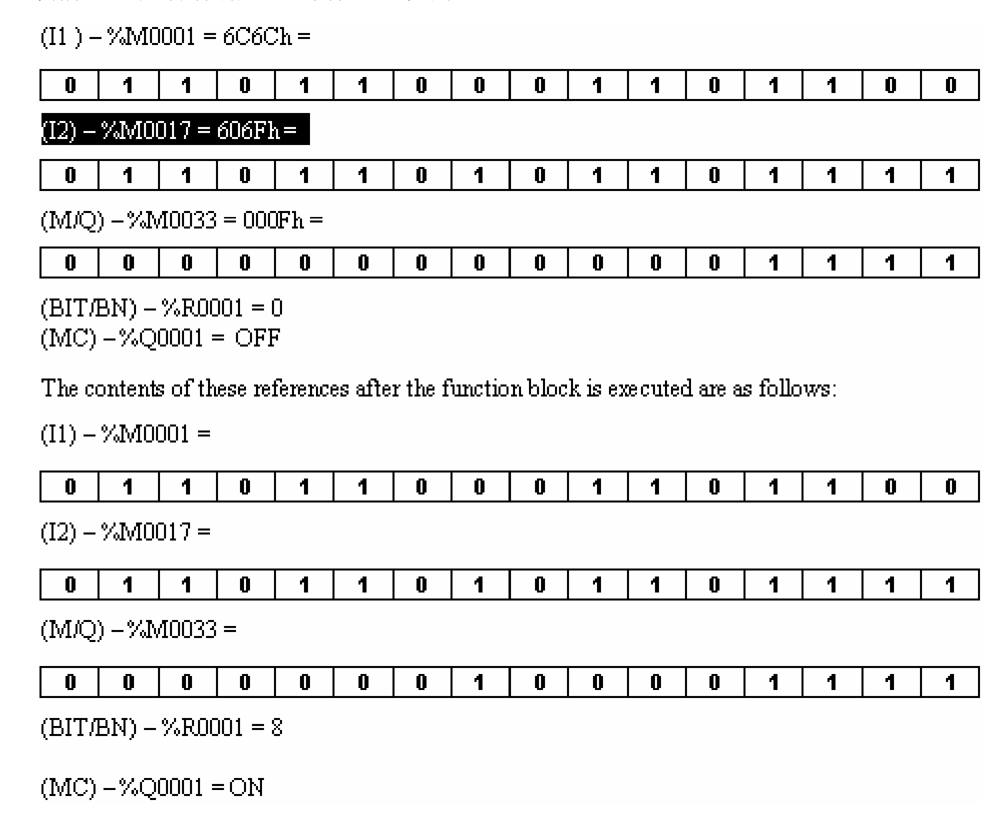
只要%I00001 被设置，MASK_COMP_WORD 将比较 VALUES 和 EXPECT 中对应的位。比较从 BITNUM+1 位开始。如果一个未被屏蔽的不可比较位被检测到，那么比较停止。屏蔽串 RESULT 中对应的位被置 1。BITNUM 被更新为不可比较位的位编号。此外，输出串 NEWVALS 更新为 RESULT 的新值，并且线圈%Q00002 接通。只要 MASK_COMP_WORD 接收到能流，线圈%Q00001 就接通。

例 2

在第一次扫描中，屏蔽比较字模块开始执行。%M0001 到%M0016 与%M0017 到%M0032 进行比较。%M0033 到%M0048 包含了屏蔽值。在%R0001 中的值确定了两个输入字串开始比较的位置。



在比较前，上面各存储器或寄存器的值为：



(I1) – %M0001 = 6C6Ch =

0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0

(I2) – %M0017 = 606Fh =

0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1

(M/Q) – %M0033 = 000Fh =

0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

(BIT/BN) – %R0001 = 0
(MC) – %Q0001 = OFF

The contents of these references after the function block is executed are as follows:

(I1) – %M0001 =

0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0

(I2) – %M0017 =

0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1

(M/Q) – %M0033 =

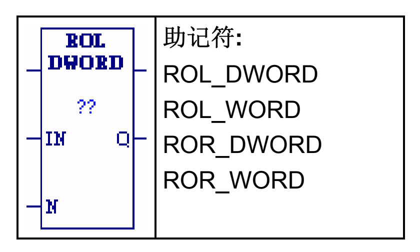
0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1

(BIT/BN) – %R0001 = 8

(MC) – %Q0001 = ON

#FST_SCN 触点强制一次且仅一次执行；否则，该模块将可能重复执行导致不可预料的结果

位循环



助记符:
ROL_DWORD
ROL_WORD
ROR_DWORD
ROR_WORD

当使能输入有效，循环右移功能模块(ROR_DWORD 和 ROR_WORD)和循环左移功能模块(ROL_DWORD 和 ROL_WORD)将分别向左循环或向右循环一个单字或双字串的所有位 N 位，指定的位数从输入字串一端移出，回到字串的另一端。

位循环功能模块向右传递能流，，除非循环位数小于 0 或者大于字串的总长度。循环结果放在输出字串 Q 里。如果想循环输入字串，输出字串 Q 的参数必须和输入字串 IN 的参数必须在相同的存储单元。在每次接收到能流的扫描中，循环后的字串被写入。

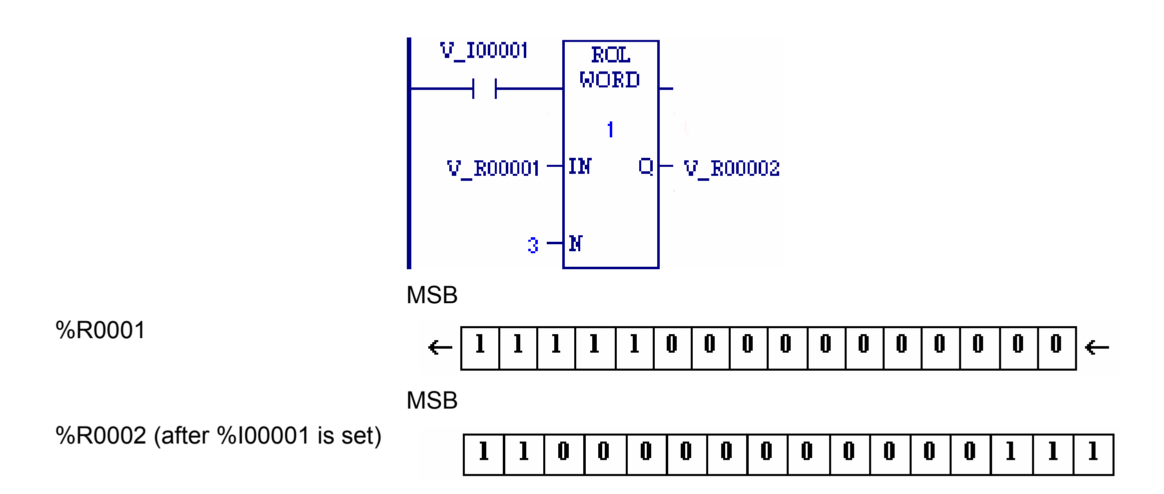
字串的长度可以指定为 1 到 256 的单字或双字。

操作数

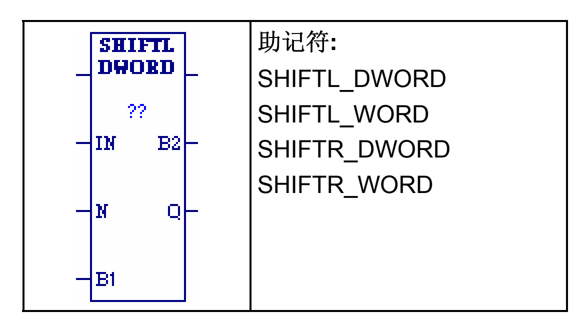
参量	描述	许用操作数	可选性
Length (??)	在循环字串里 WORD 或 DWORD 数。1 ≤ Length ≤ 256.	常数	No
IN	循环的字串	任何操作数。当长度为 1 时，常数合法	No
N	循环次数，0 ≤ N ≤ Length.	除了在%S - %SC 存储器中的外变量任何操作数	No
Q	循环后输出字串	除了常数和在%S 存储器中的外变量任何操作数	No

范例

只要 V_I0001 被置 1，在%R0001 输入的字串向左循环 3 次，并且把结果送给%R0002。而实际的输入字串%R0001 没有变化。

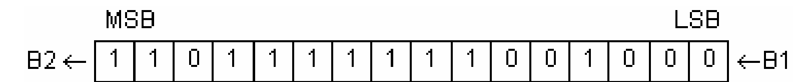


移位



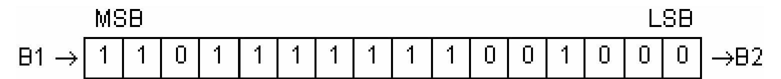
左移

当左移模块(SHIFTL_WORD)接收到能流，它将向按规定次数 N 左移一个字或一个字组里所有的位。当移位执行时，所指定的位数将向左移并移出字串的高位(MSB)，相同数量的位将移入字串的低位(LSB)。SHIFTL_DWORD 模块功能与 SHIFTL_WORD 相似。



右移

当右移模块(SHIFTR_WORD)接收到能流，它将向按规定次数 N 右移一个字或一个字组里所有的位。当移位执行时，所指定的位数将向右移并移出字串的低位(LSB)，相同数量的位将移入字串的高位(MSB)。SHIFTR_DWORD 模块功能与 SHIFTR_WORD 相似。



左移和右移

字串的长度可以指定为 1 到 256 的字。

移入字串起始的位由输入参数 B1 指定。如果 N 的值大于 1，则每个位用一样的值(0 或 1)填入。可以是：

- 其他程序模块的布尔量输出。
- 全部为 1，用#AWL_ON (始终是 ON)系统位(在存储单元%S7 内)，作为条件输入 B1。
- 全部为 0，用#AWL_OFF (始终是 OFF)系统位(存储单元%S8 内)，作为条件输入 B1。

位移位功能模块传向右传递能流，，除非移动的位数为 0 或者大于移位数据的位数。

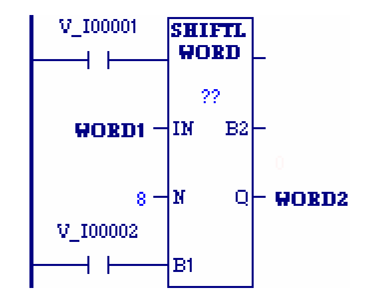
输出 Q 是移位后的输入字串的副本，如果也想移位输入字串，输出参数 Q 必须和输入参数 IN 用相同的存储单元。在每次接收到能量流的扫描中，移动后的字串被写入。输出 B2 是最后一个移出的位。举例说，如果四个位被移位，B2 则是第四个位。

操作数

参量	描述	许用操作数	可选性
Length (??)	组成字串的 WORD 或 DWORD 的数量。1 ≤ Length ≤ 256。	常数	No
IN	The string of WORDs or DWORDs to shift 要移位的字串。	任何操作数 只有当长度为 1 时，常数合法	No
N	位移动的次数，0 ≤ N ≤ Length 如果 N 为 0，字串不移动，但是生成能流。如果 N 大于位数，字串 Q 的位全部置为 B1 值，OK 被置为 FALSE，B2 置为 B1。	除了位于%S-%SC 存储器中的变量之外的所有操作数数	No
B1	移入字串的位的值	流	No 否
B2	最后一个移出字串的位的值	流	Yes 是
Q (必须和 IN 数据类型相同)	移位后字串的第一个 WORD 或 DWORD	除了常数位于%S 存储器中的变量之外的所有操作数数	No 否

范例

只要输入 V_I0001 被置 1，从 WORD1 开始的输入位串被复制到从 WORD2 开始的输出位串。根据输入值 N，WORD2 左移 8 位，输出位串开始的第一个位被置入 V_I0002 的值。



线圈

线圈常用于控制分配给它们的离散点（BOOL 型点）条件逻辑必须用来控制到线圈的能流。线圈直接驱动控制对象。线圈不传递能流。如果在程序中执行另外的逻辑作为线圈条件的结果，可以给线圈或顺延线圈/触点组合用一个内部点。

- 一个顺延线圈不使用内部点。它的后面是一个顺延触点，该触点在顺延线圈后面任一梯级的开始。
- 输出线圈总是在逻辑行的最右边。

线圈校验

线圈校验的等级通过缺省被置进“Show as error”。如果想用警告代替出错，或警告都不需要，通过编辑程序软件中 PLC 操作选项：“**Multiple Coil Use Warning** ”。

“Show as warning” 操作选项允许和多级线圈，线圈置位，线圈复位等一起使用任一线圈点。但每次操作时注意确认时间。对于“Show as warning” 和 “no warning”这两个操作选项， 一个点被置位线圈或一般线圈置为“ON”，被复位线圈或一般线圈置为“OFF”。

线圈的图形表示法

程序软件根据指定的相关 BOOL 型变量的状态分别地显示出 COIL, NCCOIL, SETCOIL, 和 RESETCOIL 指令。将在对每一种线圈的讨论时举例说明。对记忆力的讨论，查阅第 7 章中“逻辑和数据的记忆力”部分。

线圈 (常开 NO)

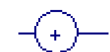


记忆型线圈表示图　　　　非记忆型线圈表示图

当一个线圈接收到能流时，置相关 BOOL 型变量为 ON（1），没有接收到能流时，置相关 BOOL 型变量为 OFF（0）。线圈可以被指定为记忆型或非记忆型变量。

有效的存储器范围%Q, %M, %T, %SA - %SC, 和%G.。允许是符号离散型变量。也允许是字导向存储器（%AI 除外）中字里的位基准，包括符号非离散型存储器。

顺延线圈



顺延线圈使 PLC 在下一级的顺延触点上延续本级梯形图逻辑能流值（TRUE 或 FALSE）。顺延线圈的能流状态传递给顺延触点。

注释:

- 如果逻辑能流在驱动顺延触点之前不驱动顺延线圈，顺延触点的状态是 FALSE。
- 顺延线圈和顺延触点不使用参量，也没有相关变量。
- 一个顺延线圈之后可以有多个含顺延触点的梯级。
- 一个含顺延触点的梯级之前可以有多个含顺延线圈的梯级。

取反线圈



记忆型取反线圈表示法 非记忆型取反线圈表示法

如果没有接收到能流，取反线圈(NCCOIL)置离散型点为 ON，如果接收到能流，取反线圈(NCCOIL)置离散型基准为 OFF。取反线圈可以被指定为记忆型或非记忆型变量.

有效的存储器范围%Q, %M, %T, %SA - %SC, 和%G.。允许是符号离散型变量。也允许是字导向存储器（%AI 除外）中字里的位基准，包括符号非离散型存储器。.

置位，复位线圈



记忆型置位，复位线圈表示法　　非记忆型置位，复位线圈表示法

置位，复位线圈常用来保持（锁存）一个点的 ON 或 OFF 的状态。

警告

置位/复位线圈给已知点写一个未定义的结果到跳变位里。通常说来，这个结果不同于由 90-70 系列 CPU 所写的，可以被后来的 PACSystems CPU 模型换掉。

给已知点写一个未定义的结果到跳变位里， 所以不要和 POSCON 或 NEGCON 的跳变触点一起使用置位、复位线圈。

当置位线圈接收到能流时，置离散型点为 ON 。当置位线圈接受不到能流时，不改变散型点的值。所以，不管线圈本身是否连续接收能流，点一直保持 ON，直到点被其他逻辑控制复位，象复位线圈等。

当复位线圈接收到能流时，置离散型点为 OFF。当复位线圈接受不到能流时，不改变散型点的值。所以，点一直保持 OFF，直到点被其他逻辑控制置位，象置位线圈等。

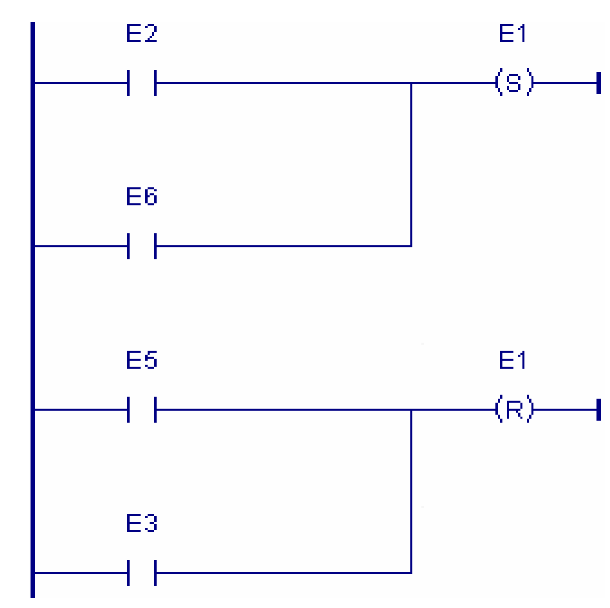
最后作用的置位线圈或复位线圈有优先权。

置位线圈或复位线圈可以是一个记忆型变量，也可以是一个非记忆型变量。

有效的存储器范围%Q, %M, %T, %SA - %SC, 和%G。允许是符号离散型变量。也允许是字导向存储器（%AI 除外）中字位点，包括符号非离散型存储器。

置位/复位线圈范例

只要点 E2 或 E6 是 ON，E1 所表示的线圈转为 ON。只要点 E5 或 E3 是 ON，E1 所表示的线圈转为 OFF。



跳变线圈

PACSystems 提供四种跳变线圈: PTCOIL, NTCOIL, POSCOIL, 和 NEGCOIL.。展示这两种跳变线圈不同作用的应用举例，看 P8-37。

POSCOIL 和 NEGCOIL



正跳变线圈 **(POSCOIL)**	*负跳变线圈* **(NEGCOIL)**
如果: ■ 变量的跳变位当前值是 OFF。 ■ 变量的状态位当前值是 OFF。 ■ 输入到线圈的能流当前值是 ON。	如果: ■ 变量的跳变位当前值是 ON, ■ 变量的状态位当前值是 OFF ■ 输入到线圈的能流当前值是 OFF,
正跳变线圈**(POSCOIL)**把关联变量的状态位转为 ON，其他任何情况下，都转为 OFF。所有的情况下，变量的跳变位都被置为能流的输入值。	*负跳变线圈*(**NEGCOIL)** 把关联变量的状态位转为 ON，其他任何情况下，都转为 OFF。所有的情况下，变量的跳变位都被置为能流的输入值。
注释： 当 **POSCOIL** 把触点的状态位转为 ON 时，也把它的跳变位为转为 ON。这就错误地描述了在 **POSCOIL 下次执行操作时**触点位转为 ON 的两个条件。这就是 **POSCOIL 下次执行操作时**触点位转为 OFF 的原因（只要触点位在此期间没有被其他逻辑控制写进状态）。	**注释：** 当 **NEGCOIL** 把触点的状态位转为 ON 时，也把它的跳变位为转为 OFF，这就错误地描述了在 **NEGCOIL 下次执行操作时**触点位转为 ON 的两个条件。这就是 **NEGCOIL 下次执行操作时**触点位转为 OFF 的原因（只要触点位在此期间没有被其他逻辑控制写进状态）

警告

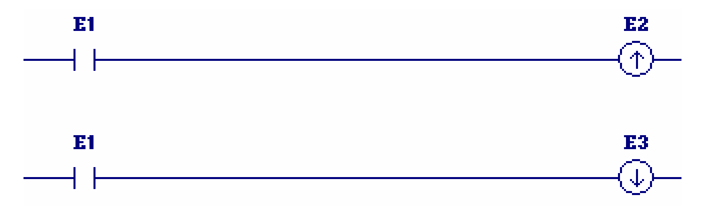
- **不要越过 POSCOIL 或 NEGCOIL 而强制触点位。如果跳变线圈被越过，然后越过又被撤除，这种在下次扫描中执行操作的跳变线圈的动作取决于很多输入，也很难推断。这将对梯形图逻辑和 CPU 的现场附属设备造成意外。**
- **如果想保持跳变线圈的单触发特性，不要用其他指令写入触点位，例如象其他的线圈或 GE 功能块。**
- **不要对跳变触点使用和跳变线圈相同的基准地址。这两种指令的相互作用很难判断。**

POSCOIL 和 NEGCOIL 的操作数

参量	*描述*	*许用操作数*	*可选性*
BOOL_V	和 POSCOIL 和 NEGCOIL 相关的变量	I, Q, M, T, G, SA, SB, SC, 和符号离型变量	No

POSCOIL 和 NEGCOIL 应用举例

当触点 E1 从 OFF 到 ON，线圈 E2 和 E3 接收到能流，每次逻辑扫描，E2 转为 ON。当 E1 从 ON 到 OFF，E2 和 E3 的能流被撤除，每次逻辑扫描，线圈 E3 转为 ON。



PTCOIL 和 NTCOIL

PTCOILs 和 NTCOILs 表现得和 POSCOIL 和 NEGCOILs 非常相似。他们之间主要的不同是：PTCOIL 和 NTCOIL 有与每个逻辑中实例线圈相关的实例数据。这个实例数据储存着线圈上一次被操作时进入线圈的能流值。在逻辑中 PTCOIL 和 NTCOIL 的 每个值都有自身实例数据的副本，因此，两个 PTCOIL 的操作是相互独立的，即使他们有相同的基准地址。

因为 PTCOIL 和 NTCOIL 发生的变化完全由进入线圈的当前能流和上次进入线圈的能流决定（例如实例数据），所以， 不受由其他线圈或逻辑指令写入与之相关的 BOOL 变量的影响。因而很多用于 POSCOIL 和 NEGCOI L 的警告不用于 PTCOIL 和 NTCOIL。.

-(P)- *正跳变线圈* **(PTCOIL)**	-(N)- *负跳变线圈* **Coil (NTCOIL)**
当输入能流是 ON，上次能流的操作结果是 OFF（也就是实例数据是 OFF），和 PTCOIL 相关的 BOOL 变量的状态位转为 ON。 在任何其他情况下，BOOL 变量的状态位转为 OFF. BOOL 变量的状态位被刷新后，PTCOIL 相关的实例数据置为输入能流的值。	当输入能流是 OFF，上次能流的操作结果是 ON（也就是实例数据是 ON），和 NTCOIL 相关的 BOOL 变量的状态位转为 ON。 在任何其他情况下，BOOL 变量的状态位转为 OFF. BOOL 变量的状态位被刷新后，NTCOIL 相关的实例数据置为输入能流的值。

PTCOIL 和 NTCOIL 的操作数

参量	*描述*	*容许操作数*	*可选性*
BOOL_V	PTCOIL 或 NTCOIL 的关联变量	在 I, Q, M, T, SA, SB, SC, or G 存储器中变量，也可以是符号离散型变量。另外在非离散型存储器（例如%R）或符号非离散型变量里的字的位触点也可以。	No

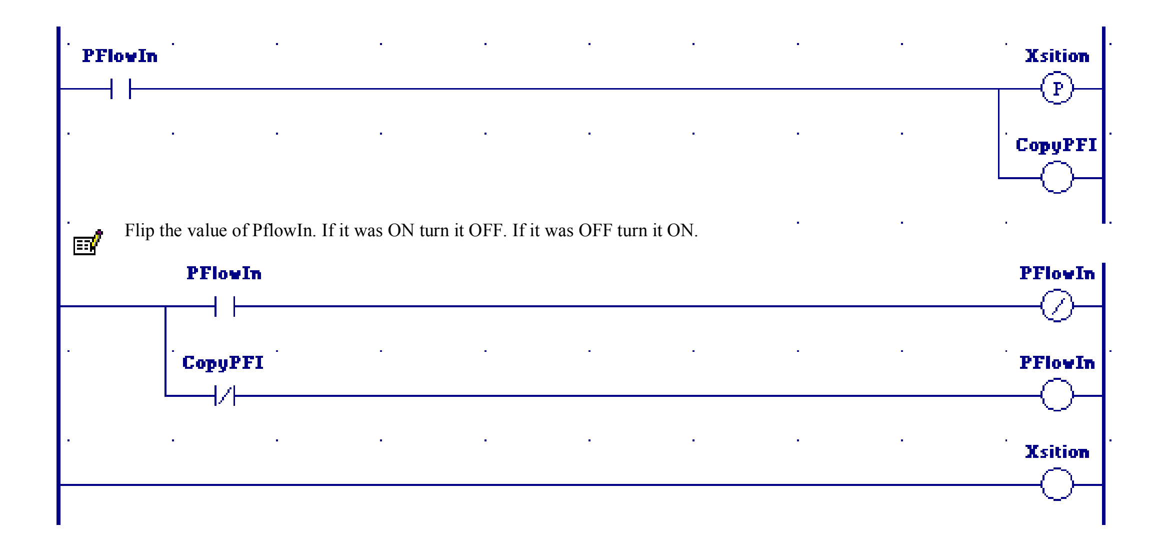
通过例子来比较 PTCOIL 和 POSCOIL

PTCOIL

在下面的离子里，进入 PTCOIL 中的能流在 OFF 和 ON 之间来回变化。第一次扫描时能流是 OFF，第二次扫描时能流是 ON，等等。每次进入 PTCOIL 的能流从 OFF 变为 ON，Xsition 的值转为 ON，因此第一次扫描时，PTCOIL 把 Xsition 转为 OFF，第二次扫描时，PTCOIL 把 Xsition 转为 ON，第三次又转为 OFF，如此这样。注意 PTCOIL 的动作不受第四梯级存在的影响，也写进 Xsition 。当第四梯级删除时，PTCOIL 动作不变。

POSCOIL

如果在下面的例子中用 POSCOIL 替换 PTCOIL（保持其余的逻辑不变，POSCOIL 的能流输入也是交替变化），逻辑行为将会不同。 POSCOIL 的状态将会受到第四次扫描过程的影响，写入 Xsition ，改变其状态和跳变位。在这个例中，POSCOIL 不会使 Xsition 转为 ON。. 如果第四梯级被删除， POSCOIL 将和 PTCOIL 的动作相同，第一次扫描时，Xsition 的值转为 OFF，第二次扫描时，Xsition 的值转为 ON，如此等等。.



触点

触点常用来监控基准地址的状态。基准地址的状态或状况及触点类型开始受到监控时，触点能否传递能流，取决进入触点的实际能流。如果基准地址的状态是 1，基准地址就是 ON；如果状态为 0，则基准地址为 OFF。

触点	*表示符号*	*助记符*	*触点向右传递能*
顺延触点	—|•|—	CONTCON	如果前面的顺延线圈置为 ON。
故障触点	BWVAR —|F|—	FAULT	如果与之相连的 BOOL 型或 WORD 变量有一个点有故障。
高位报警触点	WORDV —|HA|—	HIALR	如果与之相连的模拟（WORD）输入的高位报警位置为 ON。
低位报警触点	WORDV —|LA|—	LOALR	如果与之相连的模拟（WORD）输入的低位报警位置为 ON。
无故障触点	BWVAR —|NF|—	NOFLT	如果与之相连的 BOOL 型或 WORD 变量没有一个点有故障。
常闭触点	BOOLV —|/|—	NCCON	如果与之相连的 BOOL 型变量是 OFF。
常开触点	BOOLV —| |—	NOCON	如果与之相连的 BOOL 型变量是 ON
跳变触点	BOOLV —|↓|—	NEGCON	(负跳变触点)如果 BOOL 型输入从 ON 到 OFF。
	BOOL_V —|N|—	NTCON	(负跳变触点)如果 BOOL 型输入从 ON 到 OFF。
	BOOLV —|↑|—	POSCON	(正跳变触点) 如果 BOOL 型输入从 OFF 到 ON。
	BOOL_V —|P|—	PTCON	正跳变触点) 如果 BOOL 型输入从 OFF 到 ON。

顺延触点



- 在包含一个顺延线圈的程序块里，一个顺延触点从前次最后执行的一级开始延续梯形图逻辑。

顺延触点的能流状态和前次执行的顺延线圈的状态相同。顺延触点没有关联变量。

注意:

- 如果逻辑流在对顺延触点执行操作之前不对顺延线圈执行操作，顺延触点处于无流状态。
- 每次块开始执行时，顺延触点的状态被清除（置为无流）。
- 顺延线圈和顺延触点不使用参数，也没有与之相连的变量。
- 一个顺延线圈之后可以有多个含顺延触点的梯级。
- 一个含顺延触点的梯级之前可以有多个含顺延线圈的梯级。

故障触点



故障触点 (FAULT) 用来检测离散或模拟基准地址的故障，或定位故障（机箱、槽、总线、模块）。

- 为保证正确的模块状况指示，FAULT/NOFLT 触点使用基准地址（%I, %Q, %AI, %AQ ）。
- 为定位故障，FAULT/NOFLT 触点使用机箱、槽、总线、模块故障定位系统变量。

注意: 当一个与已知模块相连的故障从故障表中被清除时，该模块的故障指示也被清除。

- 对与 I/O 点故障报告，必须配置 HWC（Hardware Configuration ）来激活 PLC 点故障。

如果与 FAULT 相连的变量或存储单元有一个点故障，FAULT 传递能流。

操作数

参量	*描述*	*许用操作数*	*可选性*
BWVAR	与 FAULT 触点相连的变量	在%I, %Q, %AI, 和 %AQ 存储器中的变量，以及预先确定的故障定位基准地址。	No

高位/低位报警触点

WORDV —┤HA├—	WORDV —┤LA├—

高位报警触点(HIALR) 常用来检测模拟量输入的高位报警。高位/低位报警触点的使用必须在 CPU 配置时激活。

如果与模拟量输入相连的高位报警位是 ON，高位报警触点传递能流。

低位报警触点 (LOALR) 常用来检测模拟量输入的低位报警。低位报警触点的使用必须在 CPU 配置时激活。

如果与模拟量输入相连的低位报警位是 ON，低位报警触点传递能流。

操作数

参量	描述	容许操作数	可选性
WORDV	与 HIALR 或 LOALR 触点相连的变量	在 AI 和 AQ 存储器中的变量	No

无故障触点

BWVAR
—┤NF├—

无故障触点 (NOFLT) 用来检测离散或模拟基准地址的故障，或定位故障（机箱、槽、总线、模块）。如果与 NOFLT 相连的变量或存储单元有一个点故障，NOFLT 传递能流。

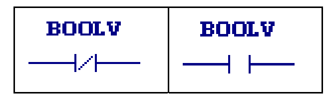
- 为保证正确的模块状况指示，FAULT/NOFLT 触点使用基准地址（%I, %Q, %AI, %AQ ）。
- 为定位故障，FAULT/NOFLT 触点使用机箱、槽、总线、模块故障定位系统变量。T
- 对与 I/O 点故障报告，必须配置 HWC（Hardware Configuration ）来激活 PLC 点故障。.

Note: 当一个与已知模块相连的故障从故障表中被清除时，该模块的故障指示也被清除。

操作数

参量r	描述	容许操作数	可选性l
BWVAR	与 NOFLT 触点相连的变量	在%I, %Q, %AI, 和 %AQ 存储器中的变量，以及预先确定的故障定位基准地址。	No

常闭触点和常开触点



如果 BOOLV 操作数是 OFF（false，0），常闭触点(NCCON) 作为一个传递能流的开关。

如果 BOOLV 操作数是 ON (true, 1)，常开触点 (NOCON) 作为一个传递能流的开关。

操作数

参量	描述	许用操作数	可选性 I
BOOLV	BOOLV 可以是一个预先确定的系统变量，或是一个自定义变量。 NCCON: 如果 BOOLV 是 ON, 常闭触点不传递能流。 如果 BOOLV 是 OFF, 常闭触点传递能流。 NOCON: 如果 BOOLV 是 ON, 常开触点传递能流。 如果 BOOLV 是 OFF, 常开触点不传递能流。	在 I, Q, M, T, S, SA, SB, SC, 和 G 存储器中的离散变量。符号离散变量，在任意非离散存储器（例如%L ）中的变量的字位基准地址，或是符号非离散变量。	No

跳变触点

从跳变触点 POSCON 和 NEGCON 输出的能流由最后写进与触点相连的 BOOL 变量决定。从跳变触点 PTCON 和 NTCON 输出的能流由与之相连的 BOOL 变量的值决定，该值是跳变触点最后一次被执行时得到的。

POSCON 和 NEGCON

BOOLV —┤↑├— *正跳变触点 POSCON*	BOOLV —┤↓├— *负跳变触点 NEGCON*
只有当下列条件全都满足时，POSCON 才向右传递能流。 ■ POSCON 的使能是 ON。 ■ 与 POSCON 相连变量的状态字当前值是 ON。 ■ 与 POSCON 相连变量的跳变字当前值是 ON 换句话说，如果有实际能流进入 POSCON，最后写进与之相连的变量值从 OFF 到 ON，POSCON 将向右传递实际能流。	只有当下列条件全都满足时，NEGCON 才向右传递能流。 ■ NEGCON 的使能是 ON。 ■ 与 NEGCON 相连变量的状态字当前值是 OFF, ■ 与 NEGCON 相连变量的跳变字当前值是 ON 换句话说，如果有实际能流进入 NEGCON，最后写进与之相连的变量值从 ON 到 OFF，NEGCON 将向右传递实际能流。

警告

不要用 POSCON 或 NEGCON 跳变触点作为跳变线圈（也叫做单触发线圈）、置位线圈和复位线圈的输入点。

■ 重要的注意事项：一旦 POSCON 或 NEGCON 触点开始传递能流，则持续到相连变量写进新的值。不管写进的值是 ON 或 OFF，POSCON 或 NEGCON 触点停止传递能流。

写进的值的来源不重要：可以是一个输出线圈，一个功能块输出，一个输入扫描，一个输入中断，一个从程序转化来的数据，或是一个外部信息。一旦新的值写进变量，相关的 POSCON 或 NEGCON 触点立即受到影响。在新的值写进变量之前，POSCON 或 NEGCOT 触点看起来被“粘住”一样。

根据逻辑流的不同，写进 POSCON 或 NEGCON 的相关变量：

■ 在一个 PLC 扫描期间可能发生多次，每一分扫描，导致 POSCON 和 NEGCON 触点变为 ON。

■ 可能分别发生在几个 PLC 扫描时间，多于一次扫描，导致 POSCON 和 NEGCON 触点变为 ON

■ 可能每次扫描发生一次，例如，如果 POSCON 或 NEGCON 的相关变量是一个%I 输入位。

一个点的越过 阻止其状态位的改变。但不阻止其跳变位的改变。如果一个越过点写进新的值，点的跳变位清零。结果是任何相关的 POSCON 和 NEGCON 触点将停止传递能流。

POSCON 和 NEGCON 的操作数

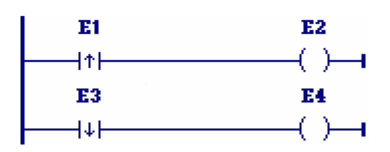
参量	*描述*	*容许操作数*	*可选性*
BOOLV	和跳变触点相连的变量	在 I, Q, M, T, S, SA, SB, SC, 和 G 存储器中的变量、符号离散变量 。	No

范例

例 1

如果变量 E1 的值从 OFF 跳到 ON，线圈 E2 转为 ON。ON 状态一直保持到 E1 再次写进新的值，E1 再次写进新的值导致 POSCON 停止传递能流。

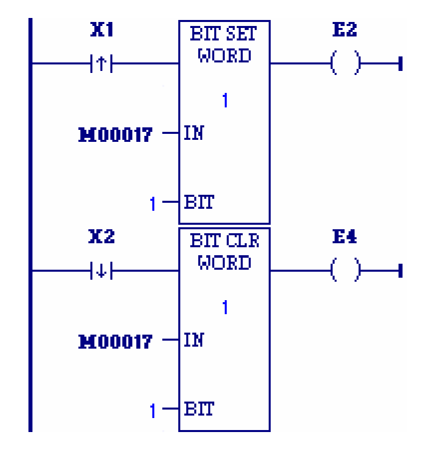
如果变量 E3 的值从 ON 跳到 OFF，线圈 E4 转为 ON。 ON 状态一直保持到 E3 再次写进新的值，E3 再次写进新的值导致 NEGCON 停止传递能流。



例 2

%M00017 位被 BIT_SET 功能块置 1，然后被 BIT_CLR 功能块清零。正跳变触点 X1 激活 BIT_SET 功能块，负跳变触点 X2 激活 BIT_CLR 功能块。

和位%M00017 相关的正跳变触点将保持在 ON 状态，直到被 BIT_CLR 功能块复位。之所以这样，是因为当 X1 从 OFF 到 ON 时，该位是只写的。与正跳变相似地，和负跳变位%M00017 相关的正跳变触点将保持在 ON 状态，直到被 BIT_SET 功能块置位。



PTCON 和 NTCON

PTCON （NTCON）触点和 POSCON （NEGCON）触点 的本质区别在于每个用于逻辑控制的 PTCON 或 NTCON 触点指令 有自己的关联实例数据。该实例数据给出了触点最后一次执行时与触点相关的 BOOL 变量的状态。因为每个 PTCON 或 NTCON 指令的实例都有自己的实例数据，使得与相同 BOOL 变量相连的两个 PTCON 或 NTCON 指令的表现不同成为可能。

BOOL_V —|P|— *正跳变触点* **PTCON**	BOOL_V —|N|— *负跳变触点* **NTCON**
只有当下列条件全都满足时，PTCON 向右传递能流： ■ PTCON 的输入使能激活。 ■ 与 PTCON 相连的 BOOL 变量的当前值是 ON。. ■ 与 PTCON 相连的实例数据的 OFF（也就是最后一次 PTCON 指令执行时关联 BOOL 变量的值是 OFF）。 这些条件满足后，控制能流，PTCON 的实例数据被刷新，BOOL 变量的当前值也被写进实例数据中。	只有当下列条件全都满足时，NTCON 向右传递能流： ■ NTCON 的输入使能激活. ■ 与 NTCON 相连的 BOOL 变量的当前值是 OFF. ■ 与 NTCON 相连的实例数据的 ON (也就是最后一次 NTCON 指令执行时关联 BOOL 变量的值是 ON). 这些条件满足后，控制能流，NTCON 的实例数据被刷新，BOOL 变量的当前值也被写进实例数据中。

重要的注意事项：一个 PTCON 或 NTCON 触点将保持 ON 状态一个“执行周期”。例如，一旦 PTCON 传递能流，下一次执行特定的 PTCON 指令时，PTCON 将不传递能流。这是因为 PTCON 的行为取决于它的实例数据值，这个值每次 PTCON 执行时就被刷新。当 PTCON 执行并传递能流时，其实例数据被刷新，刷新后的实例数据包含与之相连的 BOOL 变量的当前值，该值必须是 ON。PTCON 第二次执行时，再次传递能流的条件不满足，因为其实例数据是 ON 而不是 OFF。

这种行为和 POSCON 和 NEGCON 的行为形成鲜明的对比，它能为多个执行周期传递能流。

同时也要注意：因为 PTCON 和 NTCON 指令的行为不是由指令位决定的，这些指令可以和有些变量一起使用，这些变量在没有关联跳变位的存储器中。

PTCON 和 POSCON 的操作数

参量	*描述*	*许用操作数*	*可选性*
BOOL_V	和 PTCON 或 NTCON 相连的变量。	在存储器 I, Q, M, T, S, SA, SB, SC,和 G 中的变量，也可以是符号离散变量。或是在非离散存储器（R, AI, AQ, L, P, W）中的变量的字位以及在符号非离散变量中的字位。.	No

比较 PTCON 和 POSCON 的例子

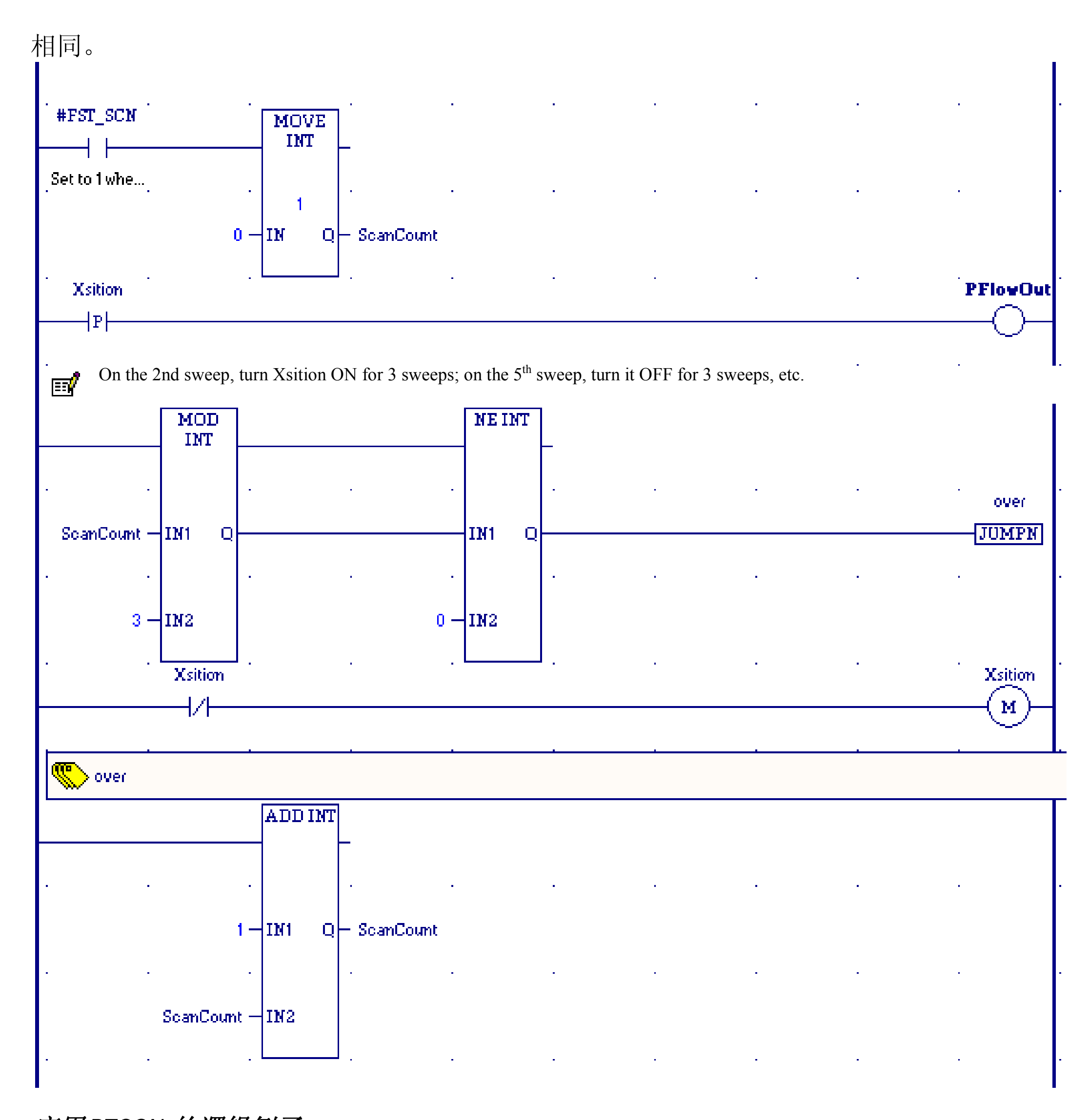
PTCON

在 P8-47 的例子里，逻辑开始执行时所有的变量置为 0。在第二次扫描开始前，用于 PTCON 的变量 Xsition 置为 1，并保持到第 2，3，4 次扫描。在第 5 次扫描之前变量 Xsition 置回 0，并保持到 5，6，7 次扫描。这样循环往复。在第 2 梯级的 PTCON 指令在第 2、8、14 次等扫描期内传递能流。前次扫描变量 Xsition 的值是 0，本次扫描变成 1。在其他的扫描中，PTCON 指令不传递能流。

POSCON

如果在 P8-47 的例子里用 POSCON 替换 PTCON（保持逻辑结构不变），变量 Xsition 的值也如同上述变化。POSCON 指令在 2、3、4 次扫描时传递能流，5、6、7 次扫描时不传递能流，8、9、10 次扫描时再次传递能流，依次类推。POSCON 的变化由变量 Xsition 跳变位的变化决定。变量 Xsition 的值写进一次，一般保持 3 个扫描期，跳变位保持相同的值 3 个扫描期。因而，POSCON 连续不断地每隔 3 个扫描周期传递能流。注意：如果变量 Xsition 的值实际上每个扫描周期就写进一次，那么 POSCON 的变化就和 PTCON 的变化

相同。



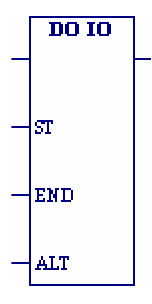
应用PTCON 的逻辑例子

控制功能

控制功能限定程序执行，改变 CPU 执行应用程序的路线。

功能	助记符	描述
Do I/O	DO_IO	一次扫描，立即刷新指定范围的输入和输出（如果 DO I/O 功能块包含模块上的所有的基准单元，模块上的所有点都被刷新，部分 I/O 模块刷新不执行）。I/O 扫描结果放在内存比放在实际输入点上好。
转鼓	DRUM	按照机械转鼓排序的式样，给一组 16 位离散输出提供预先确定的 on/off 模式。
循环	FOR_LOOP EXIT_FOR END_FOR	循环。在 FOR_LOOP 指令和 END_FOR 指令之间重复执行逻辑程序指定的次数或遇到 EXIT_FOR 指令时结束循环。
PID 控制	PID_ISA PID_IND	提高能够两个 PID 闭环控制算法： 标准 ISA PID 算法 (PID_ISA) 独立项算法 (PID_IND) **注意:** 详情查阅第 10 章.
读转换开关位置	SWITCH_POS	读 Run/Stop 转换开关的位置和转换开关配置的方式。
服务请求	SVC_REQ	请求一个特殊的 PLC 服务。. **注意:** 详情查阅第 9 章.
暂停 IO	SUS_IO	暂停一次扫描中所有正常的 I/O 刷新，DO I/O 指令指定的除外。

Do I/O



当 DO I/O (DO_IO) 接收能流，在程序运行时，每次扫描就刷新输入或输出点。除了正常的 I/O 扫描外，在程序执行期间也可以利用 DO_ IO 功能刷新其他选择的 I/O。

注意: 可以把 DO_IO 和 SUS_IO 配合使用，SUS_IO 停止正常的 I/O 扫描。

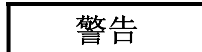
如果输入指定，DO_IO 允许程序逻辑使用输入的最新值。如果输出指定，DO_IO 根据存储在 I/O 存储器中最新值刷新输出。在整个 I/O 模块增量范围内使用 I/O；如有必要，在 DO_IO 执行期间，PLC 调整基准点。DO_IO 不扫描没有配置的 I/O 模块。

DO_IO 连续执行到所有在选择范围内的输入点全部被扫描到，或者所有 I/O 模块上的输出点已经扫描完成，然后程序返回到 DO_IO 后面的功能块。

如果模块范围包括一个选择模块（HSC，APM 等），模块的所有输入数据（%I 和%AI）或输出数据(%Q and %AQ)被扫描。在扫描选择模块时，忽略 ALT 参数。

只要接收到使能，DO_IO 向右传递能流。下列条件出现时，不传递能流：

- 不是所有指定类型的模块在选择范围之内。
- CPU 不能完全处理由功能块建立的临时 I/O 表。
- 指定模块包括和“Loss of I/O ”故障有关的 I/O 模块。



如果 DO_IO 和定时或 I/O 中断一起使用，和扫描过的输入有关的跳变触点可能不象预想的那样运行。

输入 Do I/O

当 DO_IO 接收到能流，且输入基准指定，PLC 从 ST 到 END 对输入点进行扫描。如果给 ALT 指定一个输入基准，新的输入值的副本放在从该输入基准开始的存储器中，实际输入值不刷新。ALT 必须和扫描到的输入类型大小相同。如果 ST 和 END 使用的离散输入，ALT 也必须是离散的。

如果没有给 ALT 指定输入，实际输入值就被刷新。这种情况下，允许在 CPU 扫描程序执行部分期间，扫描输入点一次或数次。

输出 Do I/O

当 DO_IO 接收到能流，且输出参考点指定，PLC 对输出点进行写操作。如果没有给 ALT 指定具体的数值，写进输出模块的输出范围由 ST 和 END 指定。如果写进输出点的输出是来自%Q 或 %AQ 之外的内存，起始参考点指定给 ALT，结束参考点由 END 到 ST 的长度自动计算。

操作数

参量	描述	许用操作数	可选性
ST	被扫描的输入或输出点（字）组起始地址。ST 和 END 必须在相同的存储区域。 **注意:** 如果 ST 和 END 在 BOOL 量存储器中，ST 必须是字节组。那就是说，基准地址必须从（8n+1）开始，例如%I01, %Q09, %Q49。	I, Q, AI, AQ	No
END	被扫描的输入或输出点的最后一位的地址。必须和 ST 在相同的存储区域。 注意：如果 ST 和 END 在 BOOL 量存储器中，END 的基准地址必须是 8n ，例如%I08, %Q16。	I, Q, AI, AQ	No
ALT	对于一个输入扫描，ALT 指定地址来存储扫描输入点/字的值。对于一个输出扫描，ALT 从指定地址获得输出点/字的值发送到 I/O 模块中。 ■ 注意：如果 ST 和 END 在模拟量存储器中，ALT 仅是一个 WORD。	I, Q, M, T, G, R, AI, AQ	Yes

范例

输入 Do I/O

当 DO_IO 使能激活，PLC 扫描%I0001-64，%Q0001 接通。 扫描过的输入点的状态信息放在内存%M0001-64。因为给 ALT 指定一个地址，不刷新实数输入。这样就允许输入的当前值和扫描开始时的值进行比较。DO_IO 的这种形式允许输入点在 CPU 执行程序期间被扫描一次或多次。.

V_I00001 DO IO V_Q00001
V_I00001 ST
V_I00064 END
V_M00001 ALT

输出 Do I/O

因为 ALT 占用一个地址，在%AQ001-004 的值写不进输出模块。当 DO_IO 使能激活，PLC 把%R0001-0004 的值写进模拟量输出模块，%Q0001 接通。

V_I00001 DO IO V_Q00001
V_AQ0001 ST
V_AQ0004 END
V_R00001 ALT

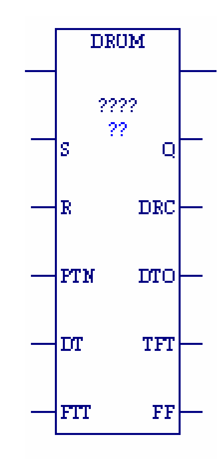
Drum

Drum 功能块象机械转鼓排序一样。转鼓排序器在一组潜在输出位组之间移步，然后选择其中基于输入的一个进功能块。选择的值被复制到一组 16 位离散的输出基准地址。

当 DRUM 功能块接收到能流，复制选择的点的内容到 Q 点中。

进入 R（复位）输入或 S（移步）输入的能流选择被复制的地址。

????（控制块）输入是转轮顺序发生器功能的参数块的起始点，参数块包含用于转鼓排序器功能的信息。



如果功能块从左侧有使能输入，并且没有错误条件存在，就向右传递能流。

转鼓排序器第一次进入新的一步，DTO (暂停超时) 位被清零 ，下列条件存在：

- DRUM 功能进入下一步，通过改变“激活步”或使用 S 输入。
- 和步数（即使是 0）相关的 DT 值是任何数

在第一次扫描时，初始化“激活步”。

操作数

参量	描述	许用操作数	可选性
????	(控制块) 一个包含转鼓排序器控制块的 5 字数组的起始地址。控制块的内容将在下面叙述。	R, P, L, W, 符号地址。	No
??	(长度)步数的指定值，在 1 到 128 之间。	常数	No
S	步数输入端。用于正向顺序进入下一步。当功能块接收到能流，S 从 OFF 到 ON 跳变，转鼓排序器移动一步。当 R 激活，功能块忽略 S。	流	No
R	复位输入端。用于选择指定顺序中的步数。当 DRUM 和 R 都接收到能流，DRUM 复制控制块中的预设步数到控制块的活动步地址里。然后，功能块复制预设步地址中的值到 Q 地址的位里。当 R 激活，功能块忽略 S。	流	No
PTN	(样本) 字组的起始地址。字数由长度（？？）操作数指定。每个字表示一步。对于控制块中活动步字的值表示设想的控制块中活动步的特值输出组合。第一个元素对应第一个活动步值，最后一个元素对应最后一个活动步值。程序软件不能自动生成字组，必须给 PTN 提供足够的内存空间。	除了常数和 S、SA-SC 数字量数据之外的任何操作数。	No

参量	*描述*	*许用操作数*	*可选性*
DT	(暂停时间)使用 DT 操作数，也必须使用 DTO 操作数，反之亦然。DT 操作数是存储器的字长的起始地址。这里的长度就是步数。每个 DT 字对应一个 PTN 字。每个字的值表示对应转鼓排序器对应步的停留时间，单位是 0.1 秒。给定步的停留时间一到， DTO 位置位。 如果暂停时间固定，转鼓排序器要等停留时间到才能进入下一步。程序软件不能自动生成字组，必须给 DT 提供足够的内存空间。	All except S, SA, SB, SC and constant	Yes
FTT	(故障超时)果使用 FTT 操作数，也必须使用 TFT 操作数，反之亦然。FTT 操作数是存储器的字长的起始地址。这里的长度就是步数。每个 FTT 字对应一个 PTN 字。每个字的值表示对应转鼓排序器对应步的故障暂停时间，单位始 0.1 秒。故障暂停时间一到，FTT 位置位。程序软件不能自动生成字组，必须给 FTT 提供足够的内存空间。	All except S, SA, SB, SC and constant	Yes
Q	存储器中一个字，包含对应当前活动步的 PTN 元素。	All except S and constant	No
DRC	(Drum 线圈)只要功能块使能激活，活动步不等于预设步，DRC 置位。	All except S	Yes
DTO	(停顿超时用 DTO 操作数，也必须使用 DT 操作数，反之亦然。当前步的暂停时间一到，DTO 被置位。	All except S and constant	Yes
TFT	(超时) 如果使用 TFT 操作数，也必须使用 FTT 操作数，反之亦然。如果 DRUM 功能块在处于一个特殊步长于该步的故障暂停时间，TFT 被置位。	All except S and constant	Yes
FF	(首位跟踪)存储器的(Length/8+1)字节起始地址，这里的 Length 就是步数。如果 MOD(Length/8+1) >0，FF 有(Length/8+1)个字节。FF 里字节的位每个位对应 PTN 的一个字。只是 FF 里字节的位任何时候只有一个是 ON，该位对应活动步的值。第一位对应活动步 1 的值，最后一个使用的位对应指定活动步的值。.	All except S and constant	Yes

转鼓排序器控制块

转鼓排序器控制块包含操作转鼓排序器的必要信息。

address	活动步
address + 1	预设步
address + 2	步控制
address + 3	定时器控制

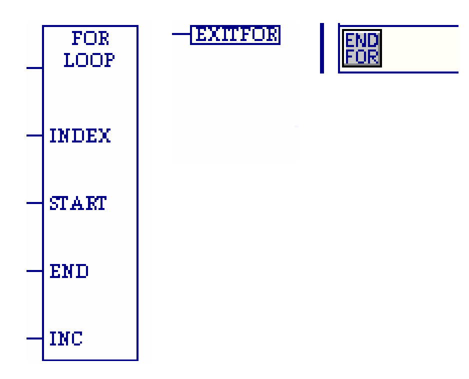
活动步 活动的值指定样本数组里的元素复制进输出存储单元。该值作为进入样本、暂停时间、故障超时、和首位跟踪操作数数组的索引使用。

预设步 当 R 时 ON 时，复制进活动步输出的一个输入字。.

步控制 用于检测 Step 输入和 Enable 输入从 **OFF** 到 **ON** 跳变的一个字。步控制字留着供功能块专用，必须不被写进。*.*

定时器控制 两个数据字，用于保存运行定时器必须的值。定时器控制留着供功能块专用，必须不被写进。

程序循环



程序循环功能循环执行梯级逻辑指定次数，同时变更 INDEX（当前循环计数）变量的值。程序循环功能开始于 FOR_LOOP 指令，结束于 END_FOR 指令。重复执行的梯级逻辑循环于 FOR_LOOP 和 END_FOR 这两个指令之间。在正常结束循环之前条件具备，EXIT_FOR 指令激活，那么就结束循环。

当 FOR_LOOP 使能激活，将保存 START（循环初值）、END（循环终值）、和 INC（增操作）操作数，计算出梯级逻辑在 FOR_LOOP 和 END_FOR 这两个指令之间循环的次数。在执行循环操作的过程中改变 START 和 END 操作数不影响操作。

当 END_FOR 使能激活, 程序循环终止，能流直接跳转到 END_FOR 指令后面的语句。

在一个梯级逻辑中，FOR 指令后面可以没有指令，FOR 指令必须是执行该梯级逻辑的最后指令。EXIT_FOR 语句只能放在一个 FOR 指令和一个 END_FOR 指令之间， END_FOR 语句后面可以没有指令或语句。END_FOR 指令也是最后执行的指令。END_FOR 语句在一个梯级逻辑中必须是唯一的。

一个 FOR_LOOP 可以通过设置一个负的增加量给下标变量递减赋值 。例如如果 START 的值 21，END 的值是 1，增加值是-5，执行 5 次程序循环，下标变量每次减少 5，下标变量的值依次是 21，16，11，6 和 1。

当 START 和 END 的值设置相等，程序循环仅执行一次。

当 START 的值不能增加/减少到 END 的值时，不执行循环语句。例如，START 的值 10，END 的值是 5，INCREMENT 的值是 1，能流直接从 FOR 语句跳转到 END_FOR 语句后面的语句。

注意: 如果在第一次测试时，FOR_LOOP 指令的使能输入有能流，在 FOR 和相应的 END_FOR 指令之间的梯级执行循环的次数由 START，END 和 INCREMENT 最初指定值决定。这种重复执行出现在 PLC 单个扫描情况，如果循环周期长， 这种重复执行可能导致看门狗定时器停止。

程序循环允许嵌套，但限制在 5 对 FOR/END_FOR 内。每个 FOR 指令后面必须有一个与之匹配的 END_FOR 指令 。

倘若是完全嵌套，也允许带有 JUMP 和 MCR 的 嵌套。MCR 和 ENDMCR 指令必须配合使用 FOR/END_FOR 指令对范围之内或之外的同一程序块中。JUMP 和 LABEL 指令也必须配合使用 FOR/END_FOR 指令对范围之内或之外的同一程序块中。跳进或跳出 FOR/END_FOR 指令对范围之内或之外的同一程序块是不允许的。

操作数

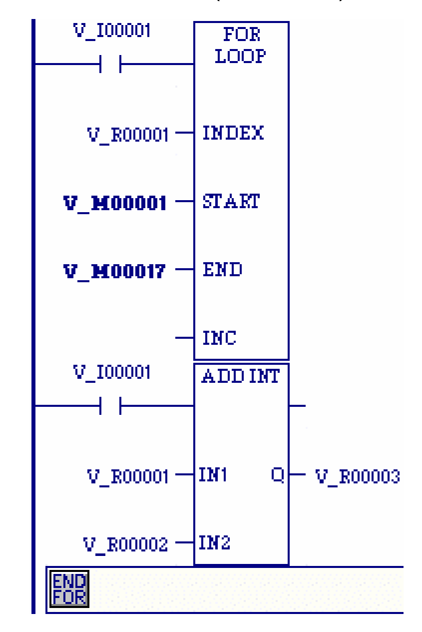
只有 FOR_LOOP 功能块需要操作数

参量	*描述*	*许用操作数*	*可选性*
INDEX	下标变量。当循环已经完成，该值未定义。 **注意:** 不推荐在循环范围内改变下标变量值。	除了常量，流和在%S - %SC 内的变量之外的任何操作数。	No
START	下标变量的开始值.	除了在%S - %SC 内的变量之外的任何操作数。	No
END	下标变量的结束值	除了在%S - %SC 内的变量之外的任何操作数	No
INC	增加值. (缺省：1.)	常量	Yes

范例

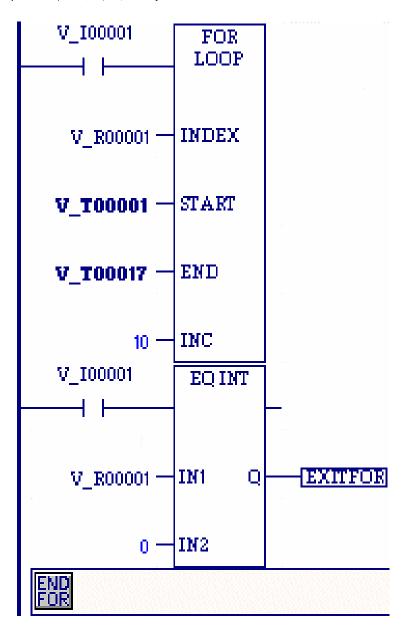
例 1

%M00001 (START)的值是 1， %M00017 (END) 的 是 10。INDEX (%R00001) 的增加值由 INC 操作数决定（缺省值 1），从 1 开始，到 10 结束。ADD 功能块执行 10 次，把 I1 (%R00001)的当前值（从 1 变到 10）加进 I2 (%R00002)的值中去。



例 2

%T00001 (START) 的值是 -100， %T00017 (END)的值是 100.。INDEX (%R00001) 的增加值是 10，从-100 开始，到+100 结束。EQ 功能块 执行 21 次，INDEX (%R00001) 的值分别等于-100，-90，-80，-70，-60，-50，-40，-30，-20，-10，0，10，20，30，40，50，60，70，80，90，100。不管怎样，当 INDEX (%R00001)为 0，EXIT 指令激活，能流直接跳转到 END_FOR 指令 之后的语句。



读转换器位置

读转换器位置(SWITCH_POS) 允许逻辑读 RUN/STOP 转换器的当前位置，以及转换器配置模式。

SWITCH POS
POS
MOD

操作数

参量	描述	许用操作数	可选性
POS	写进当前转换器位置值的存储单元： 1 - 运行 I/O 激活 2 - 运行 I/O 不激活 3 - 停止模式	除了 S, SA, SB, SC 之外任何操作数	No
MODE	写进转换器配置值的存储单元 0 – 转换器配置不支持 1 – 转换器控制运行/停止模式 2 – 转换器不用，或由拥护申请使用 3 – 转换器控制存储保护，或运行/停止模式 S	除了 S, SA, SB, SC 之外任何操作数	No

暂停 I/O



暂停 I/O (SUS_IO)功能块 在一个 CPU 扫描中出现事故时停止正常 I/O 扫描。在下次输出扫描期间，保持所有输出的当前状态。在下次输入扫描期间，输入数据刷新。在输入扫描期间，CPU 对 Genius 总线控制器完成对前次输出刷新进行校验。

注意: SUS_IO 功能暂停所有的模拟量和数字量 I/O，无任是集成 I/O、Genius I/O 或 EGD。详情查阅“ *TCP/IP Ethernet Communications for PACSystems* ”，GFK-2224。

当 SUS_IO 使能激活，所有的 I/O 服务停止，DO_IO 功能激活，恢复 I/O 服务。

警告

如果 SUS_IO 连接在梯形图左侧，没有使能逻辑控制其执行，将不断执行无规则的 I/O 扫描。

SUS_IO 只要有使能输入，就向右传递能流。

暂停 I/O 的实例

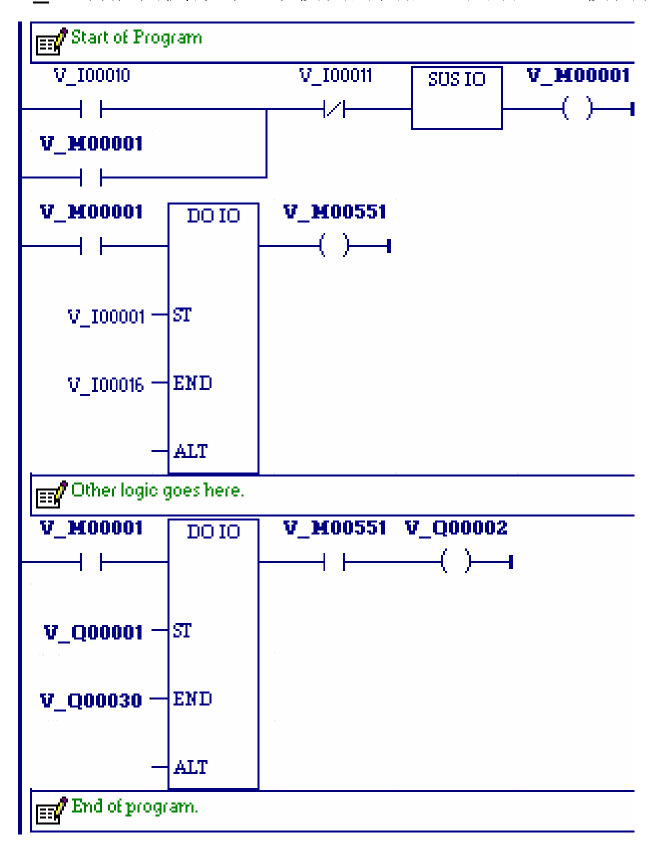
在这个例子里，一个 SUS_IO 功能块和一个 DO_IO 功能块先使程序停止 I/O 扫描，之后从程序中再进行某些 I/O 扫描。

输入 %I00010 、%I00011 和来自%M00001 的触点组成一个门闩电路。这样，每次扫描时 SUS_IO 都激活，当%I00011 为 ON 时，SUS_IO 功能不激活。如果输入点在 SUS_IO 激活后不被 DO_IO 扫描到，只有通过断 PLC 电才能使 SUS_IO 功能不激活。

SUS_IOSUS_IOSUS_IO

当两个 DO_IO 成功执行后，输出点 %Q00002 被置 1。这样的梯级构造，可以使两个功能块都执行，即使有一个的输出没有置为 1。在正常的 I/O 暂停时，输出点 %Q00002 直到和它在同一行的 DO_IO 执行时才被刷新。 这种情况在%Q00002 被置位后 CPU 扫描时才出现。一般情况下，在一个 DO_IO 功能块执行后被置位的输出点 到下一次 CPU 扫描期间另一个 DO_IO 功能块执行时才被刷新。由于有这个延时，大部分使用 DO_IO 和 SUS_IO 的程序，把 SUS_IO 功能块放在第一梯级，处理输入的 DO_IO 功能块放在下一个梯级，处理输出的 DO_IO 功能块放在最后一个梯级。

DO_IO 功能块处理输出点的范围从%Q00001 到%Q00030。如果在这个范围内的模块是一个 32 点的模块，DO_IO 功能块执行对整个模块的扫描，不会把 I/O 模块从中间分开扫描。

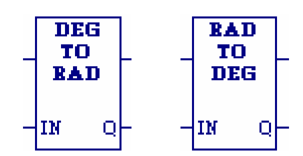


转换功能

转换功能把一个数据项目从一种数字格式（数据类型）变为另一种数字格式（数据类型）。很多程序指令，象数学函数等，必须使用一种类型的数据，因此在使用这些指令前转换数据是必要的。.

功能	助记符	描述
转换模拟量	DEG_TO_RAD	把角度转换为弧度
	RAD_TO_DEG	把弧度转换为角度
转换成 BCD4		
UINT to BCD4	UINT_TO_BCD4	把 UINT (16 位无符号整数) 转换为 BCD4
INT to BCD4	INT_TO_BCD4	把 INT (16 位带符号整数) 转换为 BCD4
把 DINT 转换为 BCD8	DINT_TO_BCD8	把 DINT (32 位带符号整数) 转换为 BCD8
转换为 INT		
BCD4 to INT	BCD4_TO_INT	把 BCD4 转换为 INT (16 位带符号整数)
UINT to INT	UINT_TO_INT	把 UINT 转换为 INT
DINT to INT	DINT_TO_INT	把 DINT 转换为 INT
REAL to INT	REAL_TO_INT	把 REAL（32 位带符号的实数或浮点数）转换为 INT
转换为 UINT		
BCD4 to UINT	BCD4_TO_UINT	把 BCD4 转换为 UINT
INT to UINT	INT_TO_UINT	把 INT 转换为 UINT
DINT to UINT	DINT_TO_UINT	把 DINT 转换为 UINT
REAL to UINT	REAL_TO_UINT	把 REAL 转换为 UINT
转换为 **DINT**		
BCD8 to DINT	BCD8_TO_DINT	把 BCD8 转换为 DINT
UINT to DINT	UINT_TO_DINT	把 UINT 转换为 DINT
INT to DINT	INT_TO_DINT	把 INT 转换为 DINT
REAL to DINT	REAL_TO_DINT	把 REAL 转换为 DINT
转换为 **REAL**		
BCD4 to REAL	BCD4_TO_REAL	把 BCD4 转换为 REAL
BCD8 to REAL	BCD8_TO_REAL	把 BCD8 转换为 REAL.
UINT to REAL	UINT_TO_REAL	把 UINT 转换为 to REAL
INT to REAL	INT_TO_REAL	把 INT 转换为 REAL
DINT to REAL	DINT_TO_REAL	把 DINT 转换为 REAL
WORD to REAL	WORD_TO_REAL	把 WORD (16 位位串) 转换为 REAL
把 REAL 转换为 **WORD**	REAL_TO_WORD	把 REAL 转换为 WORD
舍位	TRUNC_DINT	把一个 REAL 型数值通过小数部分直接舍去，保留整数部分后转换为 DINT 型数值
	TRUNC_INT	把一个 REAL 型数值通过小数部分直接舍去，保留整数部分后转换为 INT 型数值

转换模拟量



当 DEG_TO_RAD 或 RAD_TO_DEG 使能激活，对输入 IN 的值作适当的转换，把结果放在输出点 Q 中。

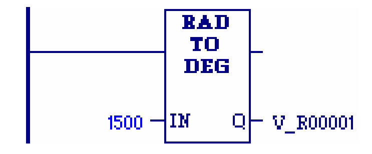
如果计算结果无溢出，DEG_TO_RAD 和 RAD_TO_DEG 向右传递能流，除非 IN 不是数字。

操作数

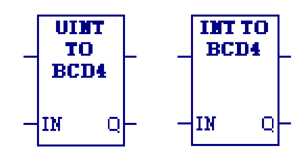
参量	描述	许用操作数	可选性
IN	被转换值.	除了 S, SA, SB, 和 SC 之外的任何操作数	No
Q	转换结果.	除了 S, SA, SB, 和 SC 之外的任何操作数	No

范例

+1500 被转换为度数，结果放在%R00001 和%R00002 里.



把 UINT 或 INT 转换为 BCD4



当功能块使能激活，把输入的 UINT 或 INT 数据转换为等效的 BCD4，并在 Q 点输出。

该功能不改变原来的输入数据。输出数据可以直接在其他程序功能块中运用。

当使能激活，功能块传递能流，除非转换结果在 0 到 9999 之外。

提示: 数据可以被转换成 BCD 格式驱动 BCD 编码 LED 显示，或者预置到外部设备中，象高速计数器等。

操作数

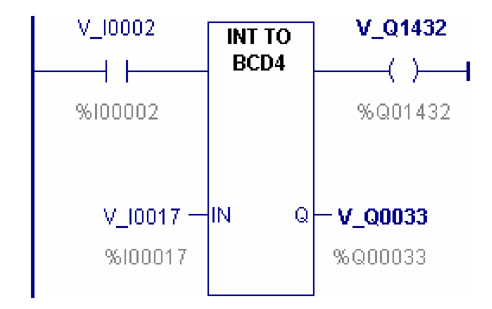
参量	描述	许用操作数	可选性
IN	转换为 BCD4 的 UINT 或 INT 值	除了 S, SA, SB, 和 SC 之外的任何操作数	No
Q	IN 中 UINT 或 INT 值的 BCD4 等效值	除了 S, SA, SB, 和 SC 之外的任何操作数	No

范例- UINT 转换为 BDC4

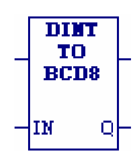
只要输入%I00002 被置位且没有错误存在，输入单元%I00017 到 %I00032 中的 UINT 被转换为 4 位 BCD 数，结果放在存储单元%Q00033 到 %Q00048 。线圈%M01432 用来校验是否成功转换。

范例 - INT 转换为 BCD4

只要输入%I00002 被置位且没有错误存在，输入单元%I0017 到 %I0032 中的 INT 被转换为 4 位 BCD 数，结果放在存储单元%Q0033 到 %Q0048 。线圈%M1432 用来校验是否成功转换。



把 DINT 转换为 BCD8



当 DINT_TO_BCD8 使能激活，把输入的 DINT（带符号的双精度整数）的值转换为等效的 BCD8 的值，并输出到 Q 中。DINT_TO_BCD8 不改变原来的 DINT 数据。

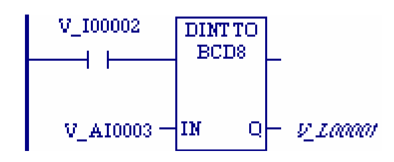
注意: 输出数据可以直接在其他程序功能块上作为输入使用。当使能激活，功能块传递能流，除非转换结果在 0 到 99,999,999 之外。

操作数

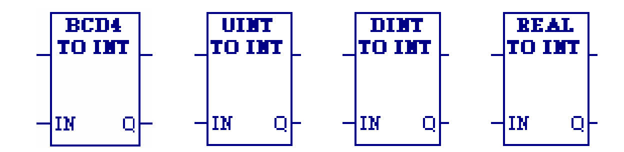
参量	描述	许用操作数	可选性
IN	转换为 BCD8 的 DINT 的值。	除了 S, SA, SB, 和 SC 之外的任何操作数	No
Q	IN 中 DINT 值的 BCD8 等效值。	除了 S, SA, SB, 和 SC 之外的任何操作数	No

范例

只要输入%I00002 被置位且没有错误存在，在输入单元%AI0003 中的 DINT 被转换为 8 位 BCD 数，结果放在存储单元%L00001 到 %L00002。.



把 BCD4, UINT, DINT,或 REAL 转换为 INT



BDC4, UINT, 和 DINT

功能块使能激活，把输入数据转换为等效的单精度带符号整数（INT），并在 Q 点输出。原始数据不被功能块改变。输出数据可以作为输入直接用于其他程序功能块中，如例所示。

当使能激活，功能块传递能流，除非数据超出范围。

REAL

当 REAL_TO_INT 使能激活，把输入的 REAL 数据按照 4 舍五入的原则转换为最接近的带符号单精度整数（INT）值，并在 Q 点输出。REAL_TO_INT 不改变 REAL 原始数据。

注意: 输出数据可以作为输入直接用于其他程序功能块中。当使能激活，功能块传递能流，除非数据超出范围或不是一个数字。

警告

从 REAL 到 INT 的转换可能导致溢出。例如，REAL 7.4E15，等于 7.4 * 10^{15},转换为 INT 时就会溢出。

提示: 把一个 REAL 值直接舍去小数部分，其余下的整数部分表示为 INT，用 TRUNC_INT 功能实现。

操作数

参量	*描述*	*许用操作数*	*可选性*
IN	转换为 INT 的值.	除了 S, SA, SB, 和 SC 之外的任何操作数	No
Q	IN 中原始输入值的 INT 等效值.	除了 S, SA, SB, 和 SC 之外的任何操作数	No

范例

BCD4 转换为 INT

只要输入%I0002 被置位，在 PARTS 中 BCD-4 的值转换为 INT，传递到 ADD_INT 功能块中，与 RUNNING 所表示的 INT 值相加，总和通过 ADD 送到 TOTAL 输出。

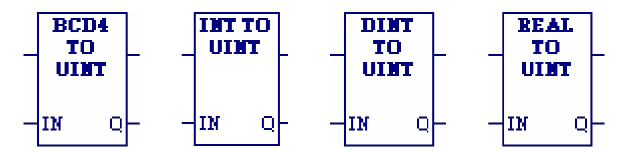
UINT 转换为 INT

只要输入%M00344 被置位，在%R00234 中的 UINT 的值被转换为 INT，传递到 ADD_INT 功能块中，与%R06488 中的 INT 值相加，总和通过 ADD 送到 CARGO 输出。

DINT 转换为 INT

只要输入%M00031 被置位，在%R00055 中的 DINT 的值被转换为 INT，传递到 ADD_INT 功能块中，与%R02345 中的 INT 值相加，总和通过 ADD 送进%R08004。

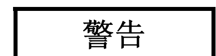
把 BCD4, INT, DINT, 或 REAL 转换为 UINT



当功能块使能激活，输入的数据被转换为等效的单精度无符号整数（UINT）值，在 Q 点输出。

到 UINT 的转换不改变原始数据。输出数据可以作为输入直接用于其他程序功能块中，如例所示。

当使能激活，功能块传递能流，除非转换结果在 0 到+65,535 范围



从 REAL 到 UINT 的转换可能导致溢出。例如，REAL 7.2E17，等于 7.2 * 10^{17},转换为 UINT 时就会溢出。

操作数

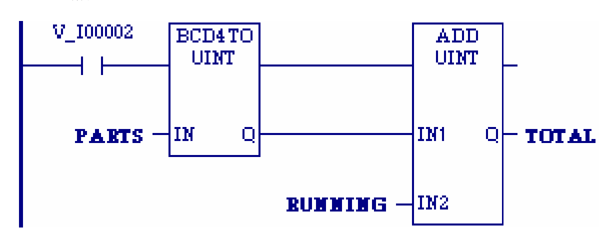
参量	*描述*	*许用操作数*	*可选性*
IN	转换为 U INT 的值.	除了 S, SA, SB, 和 SC 之外的任何操作数	No
Q	IN 中原始输入值的 UINT 等效值.	除了 S, SA, SB, 和 SC 之外的任何操作数	No

范例

BCD4 转换为 UINT

提示: BCD4_TO_UINT 的用途之一是把来自 I/O 结构的 BCD 数据转换为整数数据，并存储在存储器中。这样可以给 BCD 拨盘或 BCD 电子器件提供一个接口，象高速计数器，位置编码器等。

在下面的例子中，只要输入%I0002 被置位，在 PARTS 中 BCD4 的值被转换为一个 UINT，传递到 ADD_UINT 功能块，与 RUNNING 所表示的 UINT 值相加，总和通过 ADD_UINT 送到 TOTAL 输出。



INT 转换为 UINT

只要输入%I0002 被置位，在%L00050 中的 INT 值转换为 UINT，传递到 ADD_UINT 功能魁岸，与 %R08833 里的 UINT 相加，总和通过 ADD_UINT 送到 TOTAL 输出。

V_I00002
INT TO UINT
ADD UINT
V_L00050
IN Q
IN1 Q
TOTAL
V_R08833
IN2

DINT 转换为 UINT

只要输入%I00002 被置位，且无错误存在，在输入单元%R00007 里的 DINT 被转换为一个UINT，传送到 SUB 功能块中，减去常数 145，相减结果存储在输出存储单元%Q00033中。

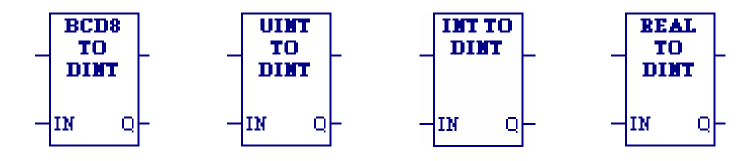
V_I00002
DINT TO UINT
SUB UINT
V_R00007
IN Q
IN1 Q
V_Q00033
145
IN2

REAL 转换为 UINT

只要输入%I00045 被置位，在%L00045 里的 REAL 值被转换为一个 UINT，传送到ADD_UINT 功能块，和%R00045 里的 UINT 值相加，总和通过 ADD_UINT 送到 TOTAL 输出。

V_I00045
REAL TO UINT
ADD UINT
V_L00045
IN Q
IN1 Q
TOTAL
V_R00045
IN2

把 BCD8, UINT, 或 INT 转换为 DINT



BCD8, UINT,和 INT

当功能块使能激活，把输入数据转换为等效的 DINT 值，在 Q 点输出。到 DINT 的转换不改变原始数据。输出数据可以作为输入直接用于其他程序功能块中。

当使能激活，功能块传递能流，除非转换结果超出范围

REAL

当 REAL_TO_DINT 功能块使能激活，把输入的 REAL 数据按照 4 舍五入的原则转换为最接近的带符号双精度整数（DINT）值并在 Q 点输出。REAL_TO_DINT 不改变 REAL 原始数据。

输出数据可以作为输入直接用于其他程序功能块中。当使能激活，功能块传递能流，除非结果超出 DINT 值的范围。



从 REAL 到 DINT 的转换可能导致溢出。例如，REAL 5.7E20，等于 5.7 * 10^{20},转换为 DINT 时就会溢出

提示: 把一个 REAL 值直接舍去小数部分，其余下的整数部分表示为 DINT，用 TRUNC_DINT 功能实现

操作数

参量	描述	许用操作数	可选性
IN	转换为 D INT 的值.	除了 S, SA, SB, 和 SC 之外的任何操作数	No
Q	IN 中原始输入值的 DINT 等效值.	除了 S, SA, SB, 和 SC 之外的任何操作数	No

范例

UINT 转换为 DINT

只要输入%M01478 被置位，在输入存储单元%R00654 里的 UINT 值转换为 DINT，结果存储在存储单元%L00049。只要功能块成功执行，输出%M00065 被置位。

BCD8 转换为 DINT

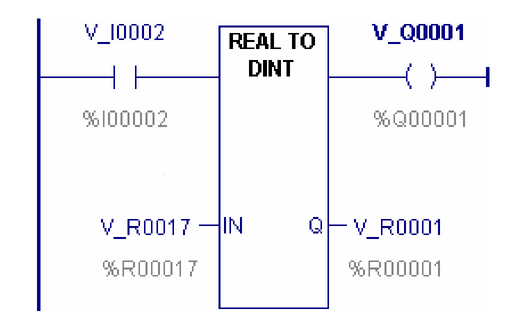
只要输入%I00025 被置位，在输入存储单元%R00654 里的 UINT 值转换为 DINT，在输入存储单元%R00797 里的 DINT 值相加。总和通过 ADD_UINT 送到 TOTAL 输出。

INT 转换为 DINT

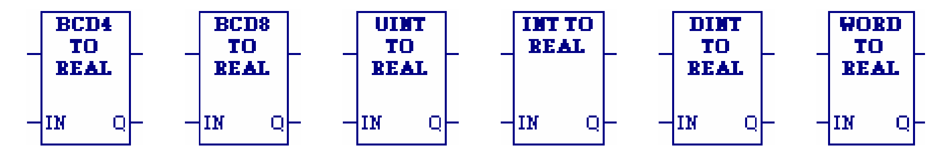
只要输入%I00002 被置位，在输入存储单元%I00017 里的 INT 值转换为 DINT，结果存储在存储单元%L00001。只要功能块成功执行，输出%Q01001 被置位

REAL 转换为 DINT

只要输入%I0002 被置位，在输入存储单元%R0017 里的 REAL 值转换为 DINT，结果存储在存储单元%R0001。只要功能块成功执行，输出%Q1001 被置位。.



把 BCD4, BCD8, UINT, INT, DINT, 和 WORD 转换为 REAL



当功能块使能激活，把输入数据转换为等效的浮点数（REAL）值，在 Q 点输出。到 REAL 的转换不改变原始数据。输出数据可以作为输入直接用于其他程序功能块中。

当使能激活，功能块传递能流，除非转换结果超出范围



把 BCD8 转换为 REAL 可能导致有效数的丢失。

这是因为一个 BCD8 值存放在一个 DWORD 中，用 32 位存放这个值，然而一个 REAL（32 位浮点数）用 8 位存放指数和符号，只有 24 位存放尾数。

警告

对于大于 7 个 10 进制有效数的数，把 DINT 转换为 REAL 可能导致有效数丢失。

这是因为一个 DINT 值要用 32 位存放，相当于 10 个 10 进制有效数。然而一个 REAL（32 位浮点数）用 8 位存放指数和符号，只有 24 位存放尾数，相当于 7 到 8 个 10 进制有效数。当这个 REAL 用一个 10 进制数显示时，可能显示到 10 位，但这些有效数是通过对 24 位尾数取舍得来的，所以至少有 2 到 3 位不精确。

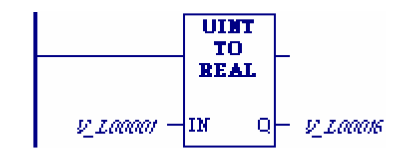
操作数

参量	*描述*	*许用操作数*	*可选性*
IN	转换为 REAL 的值.	除了 S, SA, SB, 和 SC 之外的任何操作数	
Q	IN 中原始输入值的 REAL 等效值	除了 S, SA, SB, 和 SC 之外的任何操作数	

范例

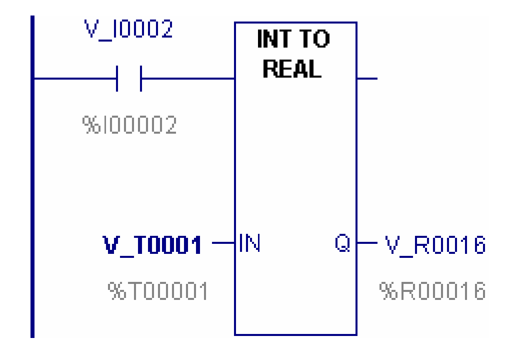
UINT 转换为 REAL

输入的 UINT 值是 825，放在存储单元%L00016 的转换结果是 825.000。

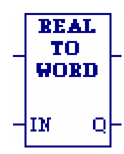


INT 转换为 REAL

输入 IN 的整数值是 678，放在存储单元%T0016 的转换结果是 678.000。



把 REAL 转换为 WORD



当 REAL_TO_WORD 使能激活，功能块把一个输入的正的 REAL 数据，按照 4 舍五入的原则转换为最接近的无符号单精度整数值，并在 Q 的 WORD 变量输出。如果转换结果是一个大于 65535 的数，只输出 65535，如果输入值是一个负数，输出为 0。REAL_TO_WORD 不改变 REAL 原始数据。

当使能激活，功能块传递能流，除非转换结果在 0 到 FFFFh 之外。

警告

从 REAL 到 WORD 的转换可能导致溢出。例如，REAL 6.8E18，等于 $6.8 * 10^{18}$,转换为 WORD 时就会溢出。

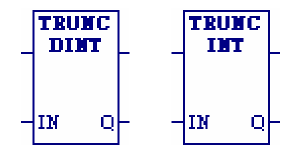
操作数

参量	*描述*	*许用操作数*	*可选性*
IN	转换为 WORD 的 REAL 值。	除了 S, SA, SB, 和 SC 之外的任何操作数	No
Q	输入的 REAL 原始数据的 WORD 等效值。0 ≤ Q ≤ 65,535.	除了 S, SA, SB, 和 SC 之外的任何操作数	No

范例

V_I00002　REAL TO WORD　RANGE WORD
V_R00001　IN　Q　V_R00003　HI_LIM　L1　Q　V_Q00001
LOW_LIM　L2
V_R00003　IN

舍位



当使能激活，舍位功能块 TRUNC_DINT 和 TRUNC_INT 分别把一个浮点数（REAL）值的小数部分直接舍去，转换为最接近的 DINT 或 INT 值。转换结果在 TRUNC_DINT 和 TRUNC_INT 功能块的 Q 点输出。原始数据不被改变。

注意: 输出数据可以作为输入直接用于其他程序功能块中。

当使能激活，功能块传递能流，除非转换结果超出范围或输入 IN 不是一个数字。

操作数

参量	*描述*	*许用操作数*	*可选性*
IN	被转换的 REAL 值，小数部分直接舍去，整数部分保持不变。	除了 S, SA, SB, 和 SC 之外的任何操作数	No
Q	IN 中 REAL 原始数值经过舍位后的值。	除了 S, SA, SB, 和 SC 之外的任何操作数	No

范例

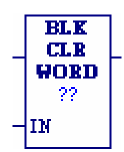
显示的常数经过舍位，所得的整数结果放在%T0001 中。

数据传送功能

数据传送功能块提供基本的数据传送功能。

功能	助记符	描述
块清零	BLK_CLR_WORD	零去替换一个块中所有的数据的值。能够被用来清零一个字的区域或是模拟存储器
块传送	BLKMOV_DINT BLKMOV_DWORD BLKMOV_INT BLKMOV_REAL BLKMOV_UINT BLKMOV_WORD	复制一个有七个常量的块到一个指定的存储单元中。这些常量是作为本功能的一部分输入的。
通信请求	COMM_REQ	允许程序跟一个智能化模块，例如一个 Genius 总线控制器或是一个高速计数器之间进行通信。
数据初始化	DATA_INIT_DINT DATA_INIT_DWORD DATA_INIT_INT DATA_INIT_REAL DATA_INIT_UINT DATA_INIT_WORD	复制一个常量数据块到一个给定范围。数据类型由助记符指定。
数据 ASCII 码初始化	DATA_INIT_ASCII	复制一个常量 ASCII 码文本块到一个给定范围
数据 DLAN 初始化	DATA_INIT_DLAN	和 DLAN 接口模块一起使用。
数据通信请求初始化	DATA_INIT_COMM	用一个常量数据块初始化一个COMM_REQ功能块。数据长度应该与COMM_REQ功能块中所有命令块。
传送数据	MOVE_BOOL MOVE_DINT MOVE_DWORD MOVE_INT MOVE_REAL MOVE_UINT MOVE_WORD	作为个别位复制数据，所以新的存储单元并不需要有相同的数据类型。数据能够被传送到一个不同的数据类型中，而不需要预先转换。
移位寄存器	SHFR_BIT SHFR_DWORD SHFR_WORD	从一个存储单元中移一个或多个数据位，数据字或数据双字到一个指定存储区域。该区域中的原有的数据被移出来了
交换	SWAP_DWORD SWAP_WORD	交换一个字数据的两个字节或一个双字数据的两个字。
总线读取	BUS_RD_BYTE BUS_RD_DWORD BUS_RD_WORD	从 VME 板中读取数据。
总线读取修改	BUS_RMW_BYTE BUS_RMW_DWORD BUS_RMW_WORD	使用 VME 总线中的读/修改/写入周期更新一个数据元素。
总线测试和设置	BUS_TS_BYTE BUS_TS_WORD	处理 VME 总线上信号量
写总线	BUS_WRT_BYTE BUS_WRT_DWORD BUS_WRT_WORD	写数据到 VME 板中

块清零



当块清零 (BLKCLR_WORD)功能块接收到能流，它就从IN开始的指定区域用零填充指定数据块。当要清零的数据来自布尔型（离散型）存储器(%I, %Q, %M, %G, or %T)时，和该区域相关的转变信息被刷新。只要BLKCLR_WORD接收到能量，就向右传递能流。

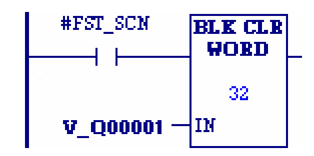
注意：线圈校验不包含输入参数IN。

操作数

参量	*描述*	*许用操作数*	*可选性*
长度（Length）(??)	从 IN 区域开始要清零的字的数量，范围在 1 到 256 之间。	常量	No
IN	存储模块中第一个要清零的单字	除了%S 存储器和数据流之外的任何操作数	No

范例

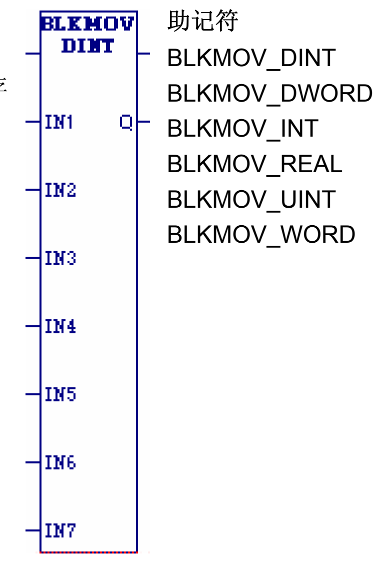
上电时，从%Q0001 开始的 32 字的%Q 存储器（512 点）都被置零。与这些区域相关的转换信息也被更新



块传送

块传送

当块传送功能块(BLKMOV))接收到能量流时，它复制一个七位常量的块到开始于输出Q中指定的目的地址的连续存储单元。只要BLKMOV功能块使能激活，就向右传递能流。



助记符

BLKMOV_DINT
BLKMOV_DWORD
BLKMOV_INT
BLKMOV_REAL
BLKMOV_UINT
BLKMOV_WORD

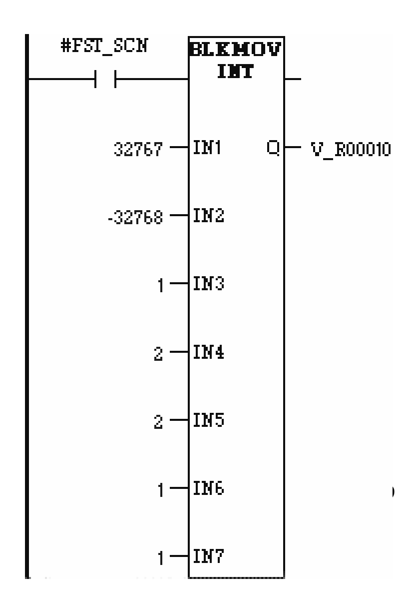
操作数

注意: 注释：对每一个助记符，对Q操作数使用相应的数据类型。例如，BLKMOV_DINT要求Q是一个DINT变量。

参量	描述	许用操作数	可选性
IN1 to IN7	传送的七个常量值	常量，必须和功能块类型匹配的常量类型	否
Q	传送数值的目的地址的第一个存储单元，IN1传送到Q	除%S外的任何操作数。%SA ， SB, SC在BLOCKMOVE REAL, BLOCK_MOVE_INT和BLOCK_MOVE_UINT也是被禁止的	否

范例

当名为#FST_SCN表示的输入使能端打开时，BLKMOV_INT把七个输入常量复制到从%R0010至%R0016的存储单元。



总线功能块

四个程序功能块允许 PACSystems CPU 跟系统中的非 GE Fanuc VME 模块之间进行通信.

- 总线读(BUS_RD)
- 总线写 (BUS_WRT)
- 总线读/修改/写 (BUS_RMW)
- 总线测试和设置 (BUS_TS)

机箱、槽、子槽、区域和偏移量参数

机箱和槽参数涉及到硬件配置中的一个模块。区域参数涉及到配置那个模块的一个存储区域。子槽一般设定为 0。偏移量是一个基于 0 的的数，用偏移量加上基地址（区域配置的的部分）计算要读和写的 VME 地址

每一个槽能够达到 8 个可配置的 VME 地址区域。这些区域能在这个模块中交迭。最少必须配置一个区域用来模块和逻辑间的通讯。VME 地址区域在模块的硬件配置存储器表中配置

一个VME综合扩展机箱的两个单宽度模块的参数

在90-70系列VME综合扩展机箱中，插槽对共享一个单独的插槽数。当两个单宽度VME模块位于这样一对插槽中时，它们具有相同的机架，插槽和子插槽参数（两个子插槽都是0）。然而，模块对中的两个模块必须设为使用不同的区域数值。例如，在插槽7的第一个模块可能使用区域数1，同样在插槽7的第二个模块可能使用区域数2。这是由插槽对进入多个区域的PACSystems配置确定。另外，每一个模块都是通过自身设置（例如，在模块中使用跳线）去响应不同的VME地址。

注意：关于在 PACSystems 系统控制系统中的选择、配置和设计非 GE Fanuc VME 模块的详细介绍，参阅 *PACSystems RX7i User'sGuide to Integration of VME Modules,* GFK-2235.

总线读

BUS_RD功能块从VME总线上读取数据。这个功能应该在程序中需要该数据之前执行。如果要被读取的数据总数在32767字节、字或双字以上，则使用多指令来读取该数据。

当BUS_RD接收到能量流，它就访问在指定机架(R)、插槽(S)、子插槽(SS)、区域地址(RGN) 和偏移量 (OFF)上的VME模块。BUS_RD从输出点Q开始在模块中复制的数据单位(双字、字或字节) 指定数量送入CPU。该功能块的操作成功时，向右传递能流。功能块执行的状态在状态位置(ST)中反映出来。

注意：对每一个 BUS_RD 功能块类型，Q 操作数类型和它是相对应的。例如，BUS_RD_BYTE 要求 Q 是一个字节变量.

注意：一个中断模块能够优先一个 BUS_RD 模块的执行。在 VME 总线上，只能连续读取 256 个字节(也就是说，没有优先执行的中断程序时读取)。

BUS RD
BYTE
??
R ST
S Q
SS
RGN
OFF

助记符:
BUS_RD_DINT
BUS_RD_DWORD
BUS_RD_WORD

总线读的操作数

参数	描述	许用操作数	可选性
Length (??)	字节、双字或字的数目，从1到32767。	常量	No
R	机箱数目，UINT常量或变量	除%S—%SC外的所有类型	No
S	插槽数目，UINT常量或变量	除%S—%SC外的所有类型	No
SS	子插槽数目（默认为0），UINT常量或变量	除%S—%SC外的所有类型	Yes
RGN	区域（默认为1），字常量或变量	除%S—%SC外的所有类型	Yes
OFF	字节偏移量，双字常量或变量	除%S—%SC外的所有类型	No
ST	操作状态，字变量	除常量和在%S—%SC中的变量外的任何操作数	Yes
Q	从VME模块读取的数据的参考，双字变量	除常量和在%S—%SC中的变量外的任何操作数	No

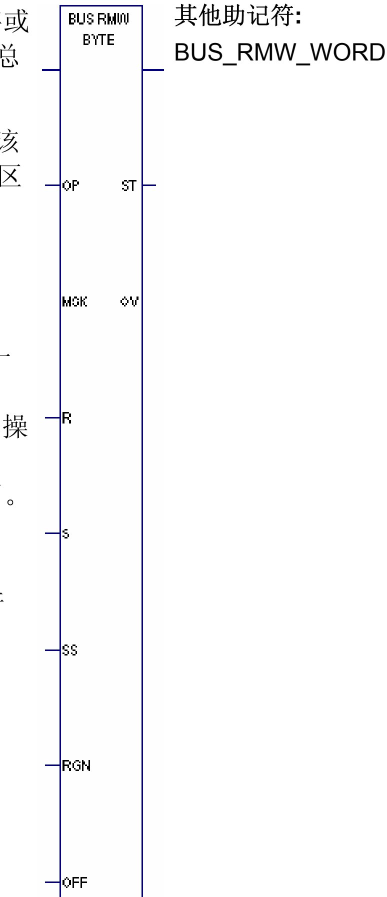
在 ST 输出中的总线读（BUS_RD）状态

读总线 BUS_RD 功能块返回下列数值中的一个到 ST 输出:

0	执行成功
1	总线出错
2	在机架或插槽位置没有模块存在
3	在该机架或插槽中的模块无效
4	开始地址超出配置地址范围
5	结束地址超出配置地址范围
6	绝对地址是偶字节，但接口配置只能是奇字节
8	区域未激活
10	功能块参数无效

总线读/修改/写

BUS_RMW 功能块更新 VME 总线上数据的一个字节、字或双字。这个功能块在执行读－修改－写操作时锁定 VME 总线.



其他助记符:
BUS_RMW_WORD

当 BUS_RMW 功能块通过它的使能输入接收能量流时，该功能块读取在指定机架(R)、插槽(S)、子插槽(SS)、随机区域地址(RGN)和偏移量(OFF)中的模块的数据中的一个双字、字或字节。初始值存在参数(OV)中。

当BUS_RMW功能块通过它的使能输入接收能量流时，该功能块读取在指定机架(R)、插槽(S)、子插槽(SS)、随机区域地址(RGN)和偏移量(OFF)中的模块的数据中的一个双字、字或字节。初始值存在参数(OV)中。

该功能块把数据和数据屏蔽(MSK)结合起来。执行的操作(AND / OR)是由OP参数选择的。屏蔽数值是双字数据。当对一个单字数据操作时，只有屏蔽数据的低16位被使用。当对一个字节数据操作时，只有屏蔽数据的低8位被使用。然后结果被重新写入读取数据的那个相同VME地址。

当操作成功时，BUS_RMW功能块向右传送能流，并返回一个状态值到ST输出。

BUS_RMW 的操作数

对于BUS_RMW_WORD，绝对的VME地址必须是2的倍数。对于BUS_RMW_DWORD，则必须是4的倍数。

绝对VME地址等于基地址加上偏移量值。

参量	*描述*	*许用操作数*	*可选性*
OP	操作类型　 0 = AND　1 = OR	常量	No
MSK	屏蔽数据，双字常量或变量	除%S—%SC的所有类型	No
R	机箱数量，UINT常量或变量	除%S—%SC的所有类型	No
S	插槽数量，UINT常量或变量	除%S—%SC的所有类型	No
SS	子插槽数目（默认为0），UINT常量或变量	除%S—%SC的所有类型	Yes
RGN	区域（默认为1），字常量或变量	除%S—%SC的所有类型	Yes
OFF	字节偏移量，双字常量或变量	除%S—%SC的所有类型	No
ST	操作状态，字变量	常量和除%S—%SC类型的变量外任何操作数	Yes
OV	初值，双字变量	常量和除%S—%SC存储器内的变量外任何操作数	Yes

ST 输出中的 BUS_RMW 状态

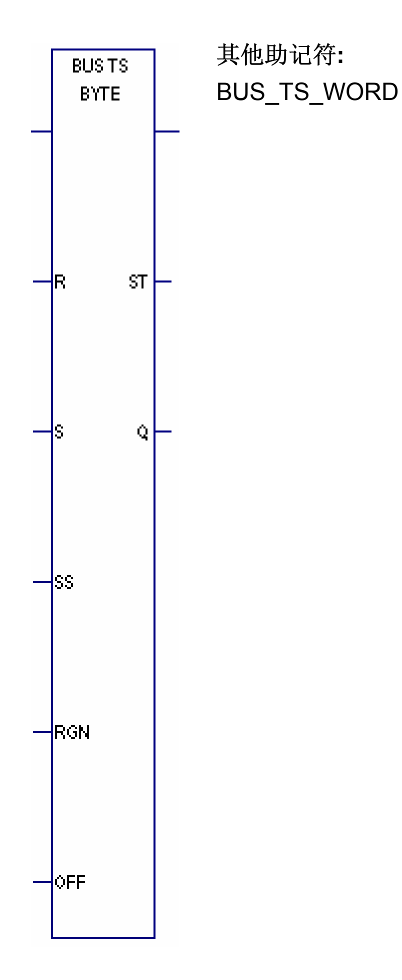
BUS_RMW 功能块返回下列数值中的一个到 ST 输出中:

0	执行成功
1	总线出错
2	模块不存在该机架或插槽位置
3	在该机架或插槽中的模块无效
4	开始地址超出配置地址范围
5	结束地址超出配置范围
6	绝对地址是偶数，但界面配置只能是奇数字节
7	对字类型，绝对VME地址不是2的倍数。对双字类型，绝对VME地址不是4的倍数
8	区域未激活
9	功能块类型对配置存取类型来说太大
10	功能块参数无效

总线测试和设置

BUS_TS 功能块处理VME总线上信号量。BUSTST功能块把一个布尔真值转换为通常在信号量存储单元上的数值。如果那个数值已经为1，BUSTST功能块就不会获得信号量。如果已经存在的数值为0，信号量被置位并且BUSTST功能块就拥有了该信号量和它控制的存储区域的使用权。通过使用BUSWRT功能块写一个0到信号位置，信号量被清零并且所有权被放弃。当执行这个操作的时候该功能块锁定VME总线。

当 BUS_TS 功能块通过它的使能输入接收能流时，该功能块将一个布尔真值跟由 RACK, SLOT, SUBSLOT, RGN, 和 OFF 参数指定的地址交换。如果信号是有用的并可以获得，功能块就将 Q 输出设为 ON。一旦使能激活并且执行时没有错误发生，它就向右传递能流.



其他助记符:
BUS_TS_WORD

总线测试和设置的操作数

BUS_TS能作为BUS_TS_BYTE 或 BUS_TS_WORD进行编程。对于BUS_TS_WORD，模块的绝对地址必须是2的倍数。对于BUS_TS_DWORD，绝对模块地址等于基地址加上偏移量值。

参量	*描述*	*许用操作数*	*可选性*
R	机箱数目，UINT常量或变量	允许的操作数	No
S	插槽数目，UINT常量或变量	除%S—%SC存储器内的变量外的所有操作数	No
SS	子插槽数目（默认为0），UINT常量或变量	除%S—%SC存储器内的变量外的所有操作数	Yes
RGN	区域（默认为1），字常量或变量	除%S—%SC外的所有类型	Yes
OFF	字节偏移量，双字常量或变量	除%S—%SC存储器内的变量外的所有操作数	No
ST	VME操作状态，字变量	除%S—%SC存储器内的变量外的所有操作数	Yes
Q	信号量是可用的，则输出 Q 设为 ON，否则 Q 设置为 OFF	常量和除%S—%SC存储器内的变量外的所有操作数	Yes

总线写

当BUS_WRT功能块通过它的使能输入接收能流时，它把位于基准地址IN的数据写入在指定机箱(R)、插槽(S)、子插槽(SS)、随机地址区域(RGN)和偏移量(OFF)的VME模块。BUSWRT写数据单元(双字、字或字节)的指定长度。该功能块的执行成功时向右传递能流。功能块执行的状态在状态位置(ST)中反映出来。

注意：对每一个BUS_WRT功能块类型，IN操作数使用的数据类型是相对应的。例如，BUS_WRT_BYTE要求IN是一个字节变量。

注意：一个中断模块能够优先一个 BUS_WRT 模块的执行。在 VME 总线上，只能连续写进 256 个字节(也就是说，没有优先执行的中断程序时写进).

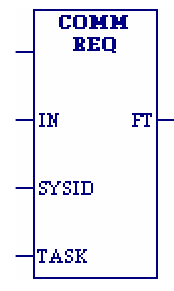
BUS WRT
BYTE
??
IN ST
R
S
SS
RGN
OFF

助记符:
BUS_WRT_DINT
BUS_WRT_DWORD
BUS_WRT_WORD

总线写的操作数

参量	描述	许用操作数	可选性
Length (??)	字节、双字或字的数目，从1到32767。	常量	No
IN	要写入VME模块的数据的基准地址，双字变量	常量和除%S—%SC存储器内的变量外所有操作数。	No
R	机箱数目，UINT常量或变量	除%S—%SC存储器内的变量外所有操作数	No
S	插槽数目，UINT常量或变量	除%S—%SC外的所有类型	No
SS	子插槽数目（默认为0），UINT常量或变量	除%S—%SC存储器内的变量外所有操作数	Yes
RGN	区域（默认为1），单字常量或变量	除%S—%SC存储器内的变量外所有操作数	Yes
OFF	字节偏移量，双字常量或变量	除%S—%SC存储器内的变量外所有操作数	No
ST	操作状态，字变量	常量和除%S—%SC存储器内的变量外所有操作数。	Yes

通信请求



通信请求(COMM_REQ)功能块用来与一个GE Fanuc智能模块通讯，例如一个Genius 通信模块或高速计数器。

注意：

■在这个部分所介绍的只是COMM_REQ功能块的基本格式。很多COMM_REQ功能块的类型都已经定义过了。对每一个不同类型的装置，需要额外的信息来编制COMM_REQ功能块的程序。

使用COMM_REQ的每一个模块的设计要求都会在专门的模块用户文档中进行描述。

- 如果使用的是串行通信，请参考12章“串行I/O,SNP和RTU协议”。
- 当一个新的内部中断具有相同的优先权时，一个正在执行的中断程序块内的COMM_REQ指令可能导致该程序块优先执行。

当 COMM_REQ 接收到能流时，它会发送由 IN 操作数指定的数据的命令块给智能或专业模块的通讯任务程序块，智能或专业模块在机箱/插槽的位置由 SYSID 操作数指定。。命令块的内容被送到接收设备并且程序执行立即重新开始。（因为 PACSystems 并不支持COMM_REQ 的等待（WAIT）模式，中断时间忽略不计。）COMM_REQ 传送能流，除非有以下的故障情况存在。有下列情况时功能块错误 Function Faulted（FT） 输出设置为ON：

- 控制块无效
- 目的地址无效（目标模块不存在或有故障）
- 目标模块队列已满，不能接收

功能块故障输出有以下状态:

Enable	Error?	Function Faulted Output
激活	否	OFF
激活	是	ON
未激活	未执行	OFF

命令块

命令块提供信息给要执行的命令的智能模块。命令块从由操作数 IN 指定的基准地址开始。这个地址可以是在内存中任何基于字的区域中(%R, %P, %L, %W, %AI, %AQ 或符号的非离散变量)。命令块的长度取决于发送到设备的数据总数。

命令块包含跟其他设备间通信的数据，加上与 COMM_REQ 执行相关的信息。利用 MOVE, BLKMOV 或 DATA_INIT_COMM 等程序功能块，可以把命令块所需要的信息放置在指定存储区域。

命令块具有下列组成：

地址	数据块的长度（字）	字数据的长度从 address＋6 开始到命令块结束。。数据块长度的范围在 1 到 128 字之间。每一个 COMM_REQ 命令都有它自己的数据块长度。当输入数据块长度时，你必须确保命令块适合寄存器的限定范围。
Address + 1	等待/不等待标志	必须设置为 0（不等待）
Address + 2	状态指针存储器类型	当 COMM_REQ 完成时，指定由设备返回的状态字所在位置的存储器类型将会被写入。见 8－87 页的“状态指针存储器类型”。
Address + 3	状态指针偏移量	Address＋3 的字包含有选择的存储器类型的状态字的偏移量。 注释：状态指针偏移量是一个基于0的值。例如，%R00001是在寄存器表中的0偏移量。
Address + 4	空闲等待时间值	该参数在不等待模式中是忽略的
Address + 5	最大通信时间	该参数在不等待模式中是忽略的
Address + 6 to Address + 133	数据块	数据块包含命令参数。数据块从 address＋6 中的一个确定要被执行的通信功能块的类型的命令数开始。参考专门的 COMM_REQ 命令格式的设备手册

状态指针存储器类型

状态指针存储器类型包含一个数字编码，这个数字编码为状态字指定用户基准存储器类型。下表给出了每一个基准存储器类型的编码:

存储器类型		*输入的十进制数*
%I	离散输入表（位模式）	70
%Q	离散输出表（位模式）	72
%I	离散输入表（字节模式）	16
%Q	离散输出表（字节模式）	18
%R	寄存器存储器	8
%W	字存储器	196
%AI	模拟量输入表	10
%AQ	模拟量输出表	12

注意:

- 输入的数决定模式。例如，如果你输入%l 位模式是 70，偏移量将会被看成是位。换句话说，如果%l 值是 16，则偏移量就被看成是字节。
- Address＋2 的高字节应该包括 0。

COMM_REQ 的操作数

参量	*描述*	*许用操作数*	*可选性*
IN	命令块的第一个字基准地址	在存储器%R, %P, %L, %AI, %AQ, %W 中的变量和非离散符号变量	No
SYSID	机箱数目（最高有效字节）和目标设备（智能模块）的插槽数目（最低有效位） **注意：** 对没有扩展机架的系统，主机箱的 SYSID 必须是零	除流和在存储器%S - %SC 中的变量外任何操作数	No
TASK	目标设备上的进程的任务 ID	常量，在存储器%R, %P, %L, %AI, %AQ, %W 中的变量和非离散符号变量	No
FT	■ 这是一个等待模式 COMM_REQ，CPU 不支持 ■ 指定目标地址(SYSID 操作数)不存在 ■ 指定任务（TASK 操作数）对设备无效 ■ 数据长度为 0 ■ 设备状态指针地址（命令块的一部分）不存在。这可能是由于选择错误的存储器类型，或存储器类型的地址超出范围	能流	Yes

COMM_REQ 举例

例 1

当使能输入%M0020 为 ON 时，在%R0016 的命令块开始被送到 PLC 中处于机架 1 插槽 2 的设备的通信任务 1 中。如果处理 COMM_REQ 时有错误发生，%Q0100 被置位。

V_M00020
COMM REQ
V_Q00010
V_R00016 IN FT
102 SYSID
1 TASK

例 2

MOVE 功能块在可以在在例 1 中描述的 COMM_REQ 功能的命令块中使用。

#FST_SCN
1
MOVE UINT 1 100 IN Q R00016
MOVE UINT 1 0 IN Q R00017
MOVE UINT 1 8 IN Q R00018
MOVE UINT 1 512 IN Q R00019
M00020
2
COMM REQ
R00016 IN FT
102 SYSID
1 TASK
Q00100

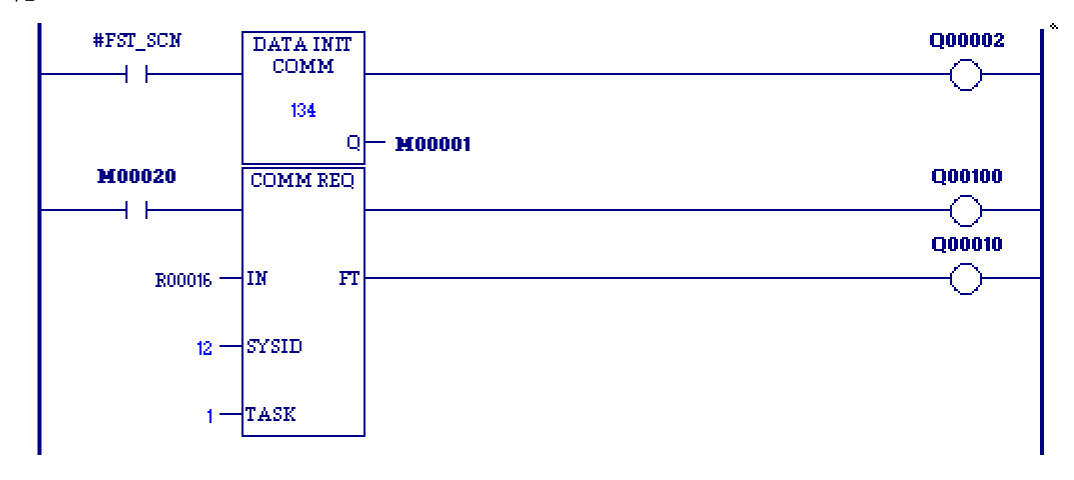
COMM-REQ 的输入 IN 指定%R00016 作为命令块的开始基准地址。连续的基准地址包含有以下内容：

%R00016	数据块长度
%R00017	等待/不等待标志
%R00018	状态指针存储器类型
%R00019	状态指针偏移量
%R00020	等待超时值(因为这个参数在不等待模式中是忽略的，没有值输入)
%R00021	最大通信时间值(因为这个参数在不等待模式中是忽略的，没有值输入)
%R00022 到数据结束	数据块

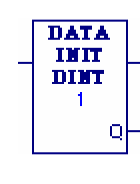
MOVE 功能为 COMM_REQ 提供以下的命令块数据

- 第一个 MOVE 功能块把正在通信等待数据的长度置于%R00016
- 第二个 MOVE 功能块把常量 0 置于%R00017。这指定了不等待模式。
- 第三个 MOVE 功能块把常量 8 置于%R00018。这指定寄存器表作为状态指针的存储单元。
- 第四个 MOVE 功能把常量 512 置于基准地址%R00019。因此，状态指针位于%R00513。

在例 2 中的程序逻辑能通过用一个 DATA_INIT_COMM 功能块取代六个 MOVE 功能块来简化。



数据初始化



助记符:
DATA_INIT_DWORD
DATA_INIT_DWORD
DATA_INIT_INT
DATA_INIT_UINT
DATA_INIT_REAL
DATA_INIT_WORD

注意：助记符 DATA_INIT_ASCII (8-错误！未定义书签。页)和 DATA_INIT_COMM (8-错误！未定义书签。页)不同于其它六个功能块。

数据初始化功能块(DATA_INIT)复制一个常量数据块到一个参考范围。

.当 DATA_INIT 指令第一次被编程的时候，常量被初始化为 0。指定要复制的常量数据，双击 LD 编辑器中的 DATA_INIT 指令。

当 DATA_INIT 接收能流时，它把常量数据复制到输出 Q。

ATA_INIT 常量数据长度指定了有多少功能块类型等待常量数据被复制到从输出 Q 开始的连续基准地址。一旦 DATA_INIT 使能激活它就向右传送能量到

注意:

- 输出参数不包含在线圈校验中。
- 如果你用另一个指令(除 DATA_INIT_ASCII 或 DATA_INIT_COMM)取代 DATA_INIT 指令 (除 DATA_INIT_ASCII 或 DATA_INIT_COMM)中的一个，逻辑 Developer - PLC 将努力保持数据的一致。例如，用 8 行配置一个 DATA_INIT_INT 并用一个 DATA_INIT_DINT 取代该指令，将会保持数据还是 8 行的。当取代 DATA_INIT_指令时，精确会下降，并且当这种情况被检测到的时候会显示一个报警信息。

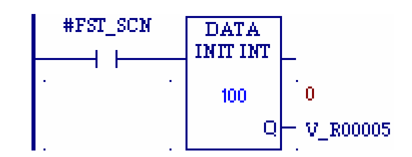
操作数

注意： 对每个助记符， Q 操作数使用相应的数据类型。例如，DATA_INIT_DINT 要求 Q 是一个 DINT 变量。

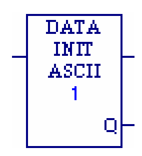
参数	*描述*	*许用操作数*	*可选性*
Length “	复制到从输出 Q 开始的连续基准地址的常量数据的数量(默认为 1)	常量	No
Q	复制数据进入区域的开始地址	除%S. SA, SB 外的所有操作数，并且 SC 对 REAL,INT 和 UINT 形式是不允许的。	No

范例

在第一次扫描(受#FST_SCN 系统变量限制)，原始数据中的 100 个通过%R00104 复制到%R00105。



数据初始化 ASCII



数据初始化 ASCII(DATA_INIT_ASCII)功能块把一个常量 ASCII 文档块复制到一个给定范围。

.当 DATA_INIT_ASCII 第一次编程时，常量都被初始化为 0。指定要复制的常量数据，双击 LD 编辑器中的 DATA_INIT_ASCII 指令。

当 DATA_INIT_ASCII 接收能流时，它把常量数据复制到输出 Q。DATA_INIT_ASCII 常量数据长度（LEN）指定了有多少常数文本的字节被复制到从输出 Q 开始的连续基准地址。LEN 必须为一个偶数。一旦 DATA_INIT_ASCII 使能激活，它就向右传送能流。

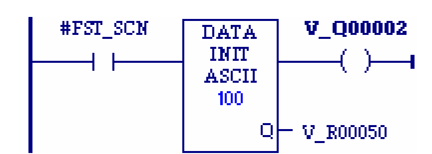
注意：输出参数不包含在线圈校验中。

操作数

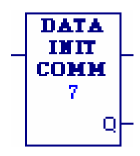
参数	描述	许用操作数	可选择
Length	复制到从输出 Q 开始的连续基准地址的常量文档的字节数目(默认为 1)。LEN 必须为一个偶数。	常量	No
Q	复制数据进入区域的开始地址	除%S 外的所有操作数	No

范例

在第一次扫描(受#FST_SCN 系统变量限制)，ASCII 文本中的 100 个字节的十进制等效值通过%R001049 复制到%R001050。%Q00002 接收使能。



数据初始化通信请求



数据初始化通信请求功能块(DATA_INIT_COMM)用一个常量数据块初始化一个 COMM_REQ 功能块。COMM_REQ 的 IN 参数必须与这个 DATA_INIT_COMM 功能块的输出 Q 相应。

当 DATA_INIT_COMM 第一次编程时，常量都被初始化为 0。指定要复制的常量数据，双击 LD 编辑器中的 DATA_INIT_COMM 指令。

当 DATA_INIT_COMM 接收能流时，它把常量数据复制到输出 Q。DATA_INIT_COMM 常量数据长度操作数指定了有多少常量数据字被复制到从输出 Q 开始的连续基准地址。长度应该等于 COMM_REQ 功能块整个命令块的大小。

一旦 DATA_INIT_COMM 使能激活，它就向右传送能流

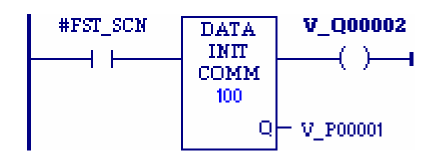
注意：输出参数不包含在线圈校验中。

操作数

参数	描述	许用操作数	可选性
Length	复制到从输出 Q 开始的连续基准地址的常量文档的字数目(默认为 7)，必须等于 COMM_REQ 功能块整个命令块的大小，包括标题(0-5 个字)。	常量	No
Q	复制数据进入区域的开始地址	R,W,P,L,AL,AQ 和符号非离散变量	No

范例

在第一次扫描(受#FST_SCN 系统变量限制)，一个由 100 个数据字组成的命令块，包括 6 个标题字，被复制进入从%P00001 到% P00100。%Q00002 接收能量。



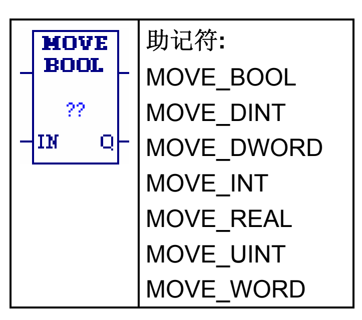
数据初始化

.数据初始化 DLAN 功能块(DATA_INIT_DLAN)和一个 DLAN 接口模块使用，该接口为一个可用性受限制的专门系统。如果你有一个 DLAN 系统，详情请查阅 *DLAN/DLAN+ Interface Module User's Manual,* GFK-0729。

操作数

参数	*描述*	*许用操作数*	*可选性*
Q	复制数据进入区域的开始地址	流， R, W, P, L, AI, AQ, 和符号非离散型变量	No

数据传送

	助记符: MOVE_BOOL MOVE_DINT MOVE_DWORD MOVE_INT MOVE_REAL MOVE_UINT MOVE_WORD

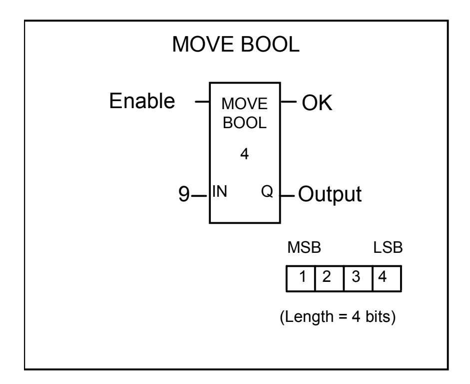
当 MOVE 功能块接收能流，它把 PLC 存储器的一个存储单元的数据作为独立位复制到另一个存储单元。因为数据是以位形式复制的，新的存储单元并不需要与源地址具有相同的数据类型。

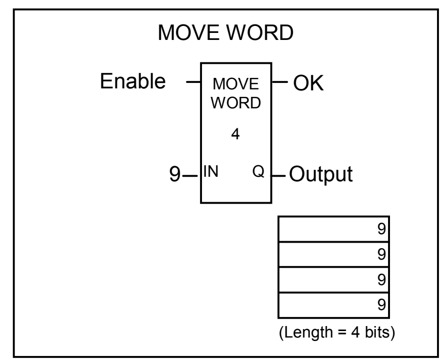
当 MOVE 功能块接收能流，其以位的形式把数据从输入操作数 IN 复制到输出操作数 Q.如果数据是从布尔(离散)存储器中一个存储单元传送到另一存储单元，例如，从%I 存储器到%T 存储器，与布尔存储器组成元素相关的转移信息被更新以表明 MOVE 操作是否导致任何布尔存储器组成元素的状态改变。在输入操作数中的数据并不改变，除非在源操作数和目的操作数之间有重叠，

注意: 如果在 Q 操作数中指定的布尔类型数据组不包括一个字节中的所有的位，当 Move 功能块接收能量流时，与那个字节(不在数组中的)相关联的跳变位被清零。输入 IN 是一个给要传送的数据提供参考地址或是一个常数地址。

如果一个常量指定了，那么该常量的值位于输出基准指定的存储单元。例如，如果一个值为 4 的常量指定给 IN，那么 4 就放在由 Q 指定的存储单元中。如果长度大于 1 并且指定了一个常量，该常量就放在由 Q 指定的存储单元开始的连续的存储单元中，存储单元数由长度（length）指定。IN 和 Q 操作数不允许重叠。

MOVE 的结果取决于为功能块选择的数据类型，如下面所示。例如，如果常量值 9 指定给 IN，长度为 4，那么 9 就放在 Q 指定的位存储单元和接下来的三个单元。





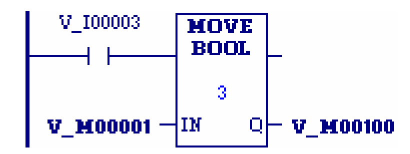
MOVE 功能块一旦使能激活就向右传递能流。

数据传送操作数

参量	*描述*	*许用操作数*	*可选性*
Length (??)	IN 的长度；要传送的位、字、双字的数目。 如果 IN 是一个常量，Q 是个布尔量，那么 1 ≤ Length ≤ 16; 否则 1 ≤ Length ≤ 256.。1 ≤ Length ≤ 32,767	常量	No
IN	第一个传送数据项的位置。 对 MOVE_BOOL，任何离散参考地址都可使用。它不需要排成字节组。从指定参考地址开始的 16 个位在线显示。 . 如果 IN 是一个常量，它就被作为一个位组处理。最低有效位的值被复制到 Q 指定的存储器位置。如果 长度大于 1，位从最低有效位到最高有效位按顺序被复制到连续的存储单元，存储单元数由长度（length）指定。	任何操作数。S, SA, SB, SC 只在字、双字、布尔类型中允许	No
Q	第一个目的地址项的位置 对 MOVE_BOOL，任何离散参考地址都可使用。它不需要排成字节组。从指定参考地址开始的 16 个位在线显示。	除%S 外任何操作数。%SA, SB, SC 也只在字、双字和布尔类型中被允许	

Example 1

一旦%I00003 被置位，%M00001, %M00002, and %M00003 三位分别被传送到 M00100, %M00101 和%M00102。线圈%Q00001 打开。

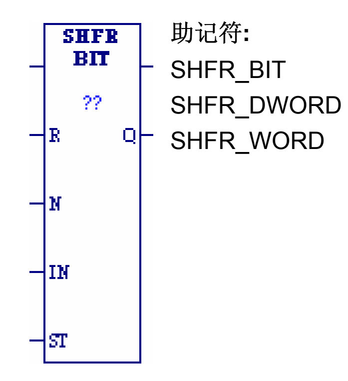


Example 2

V_M00001 和 V_M00033 都是长度为 3 的字组，在每一个组中总共有 48 位。因为 PLC 不支持数组，作为要传送的字的总数目长度必须设置为 3。当使能输入端 V_Q0014 为 ON 时，MOVE_WORD 从存储单元%M00001 传送 48 位到单元%M00033。即使目的地址与源地址重叠了 16 位，传送也是正确的。

V_Q00014
MOVE
WORD
3
V_M00001
IN
Q
V_M00033

移位寄存器



助记符:
SHFR_BIT
SHFR_DWORD
SHFR_WORD

当移位寄存器(SHFR_BIT, SHFR_DWORD 或 SHFR_WORD)接收能量，R 操作数不接收时，移位寄存器从一个基准存储单元传送一个或多个的数据位、数据双字或数据字到一个指定存储区域。一个存储器的一个邻近部分作为移位寄存器使用。例如，一个字可能移到一个有指定长度为 5 个字的存储器区域。这样移位结果是，另外的数据字将移出该存储器区域末端。

警告

在多字功能块中不推荐使用输入和输出参考地址范围重叠，因为可能产生不可预料的结果。

.复位输入 R 优先于功能块使能输入，当复位激活时，所有从移位寄存器开始的等于指定长度的基准地址都被 0 填充。

如果功能块模块接收能流并且 R 没有激活，移位寄存器的每一位、字或双字被传送到下一个最高地址。移位寄存器中的最后一个元素被移到 Q。IN 的移位寄存器元素的最高地址移到 ST 开头空出来的元素位置。

注意: 移位寄存器的内容在整个程序中都是易获得的，因为它们在逻辑可寻址寄存器中覆盖在绝对地址

一旦功能块模块使能激活而 R 不激活，其就向右传递能流。

操作数

参量	描述	许用操作数	可选性
Length (??)	移位寄存器中的数据项数目 $1 \le \text{Length} \le 256$		No
R	复位。R 为 ON 时，位于 ST 的移位寄存器用 0 填充。	能流	No
N	移进移位寄存器的数据项数目	常量	No
IN	移进移位寄存器的第一个数据项的值 **SHFR_BIT:**对%I, %Q, %M 和%T 寄存器可使用任何布尔给定，不需要按字节排序。然而，由指定的参考地址开始的 1 位在线显示。	任何操作数 位无数据流	No
ST	移位寄存器的第一个数据项 .注意：对%I, %Q, %M 和%T 寄存器可使用任何布尔给定，不需要按字节排序。然而，由指定的参考地址开始的 16 位在线显示。	除数据流，常量，S 外任何操作数	No 否
Q	T 移出移位寄存器的数据。与移进数据项相同数目的数据项将被移位出去。 **SHFR_BIT:** 对%I, %Q, %M 和%T 寄存器可使用任何布尔给定，不需要按字节排序。然而，由指定的参考地址开始的 1 位在线显示。	除数 S 外任何操作数 位无数据流	No 否

范例

SHFR_WORD 对从%R0001 到%R0100 的寄存器存储单元执行操作。当复位触点 CLEAR 激活时，移位寄存器字设置为 0。

当 NXT_CYC 触点激活而 CLEAR 没有激活，位于%Q0033 的输出状态表的字移位到%R0001 的移位寄存器。从%R0001 移位寄存器中移位出来的字存储在输出%M0005 中。注意，在这个例子里，LEN 指定的长度，移位的数据总数(N)并不相同。

NXT_CYC
SHFR WORD
CLEAR
100
R
Q
V_M00005
1
N
V_Q00033
IN
V_R00001
ST

交换

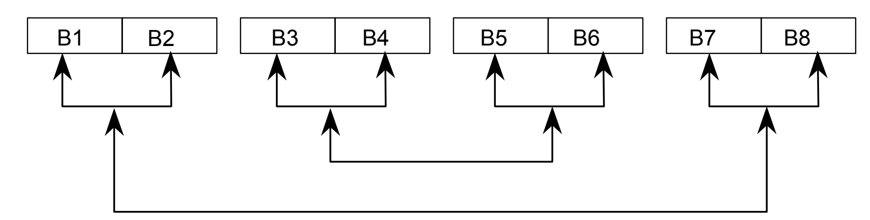


交换功能块模块被用来交换一个字中的两个字节(SWAP WORD)或是交换一个双字中的两个字(SWAP DWORD)。SWAP 可以在一个长度大于 1 的大范围存储器中执行。如果完成了，指定长度的每个字或双字中的数据被交换

当 SWAP 功能块模块接收能流，其交换基准 IN 中的数据并且把交换的数据放入输出基准 Q。该模块一旦使能激活九向右传递能流。

PACSystems CPU 按 Intel 协定以字节形式存储字数据。把一个字的最低有效字节存储在地址 n，最高有效字节存储在地址 n+1。很多 VME 模块遵循 Motorola 的协定把最高有效字节存在地址 n，最低有效字节存储在地址 n+1。

PACSystems CPU 分配字节地址 1 给相同的存储单元，不管其它设备使用的字节存储方式。然而，由于字节有效位、 字和多字数据的不同，例如，16 位整型(INT, UINT)，32 位整型(DINT)或浮点(REAL)数，在与遵循 Motorola 协定模块转换时必须调整。在这些情况下，每个字中的两个字节在转换前或转换后必须交换。另外，对多字数据项，以字为基础进行两端位置颠倒交换。例如，一个来自 Motorola 协定模块，转移到 PACSystems CPU 的 64 位实数必须被交换字节和字调换，读前后一样。如下图所示：



字符串(ASCII)或 BCD 数据不需要调整因为 Intel 和 Motorola 协定使用同样的方式存储字符串。

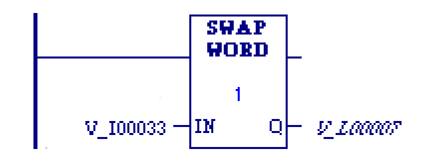
Swap 的操作数

两个参数 IN 和 Q 必须具有相同的类型，字或双字

参量	*描述*	*许用操作数 s*	*可选性*
Length (??)	操作的字或双字的数目 1 ≤ Length ≤ 256.	常量	No
IN	要交换的数据的基准地址(类型必须与 Q 相同)	任何操作数	No
Q	交换后数据的基准地址((类型必须与 IN 相同)	除 S 外任何操作数	No

Swap 应用举例

T 位于位%I00033 到%I00048 的两个字节被交换。结果存储在%L00007。

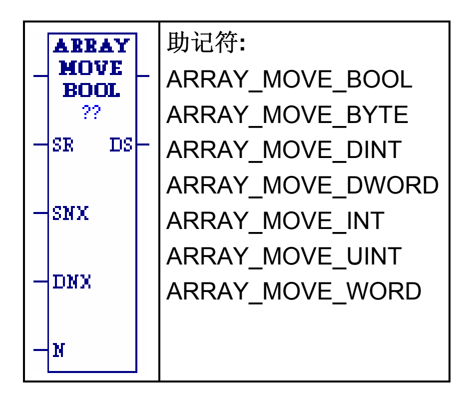


数据表功能

功能	记符	描述
数组传送	ARRAY_MOVE_BOOL ARRAY_MOVE_BYTE ARRAY_MOVE_DINT ARRAY_MOVE_INT ARRAY_MOVE_WORD	从源存储器块中复制一个给定数目的数据元素到目的存储器块中 注意：存储器块不需要被定义为数组。必须提供一个开始地址和用于传送的相邻寄存器数目。
数组范围	ARRAY_RANGE_DINT ARRAY_RANGE_DWORD ARRAY_RANGE_INT ARRAY_RANGE_UINT ARRAY_RANGE_WORD	决定一个值是否在两个表指定范围之内
FIFO 读	FIFO_RD_DINT FIFO_RD_DWORD FIFO_RD_INT FIFO_RD_UINT FIFO_RD_WORD	把位于 FIFO(先进先出)表底部的入口数据移走，指针值减 1
FIFO 写	FIFO_WRT_DINT FIFO_WRT_DWORD FIFO_WRT_INT FIFO_WRT_UINT FIFO_WRT_WORD	指针值增 1，写数据到 FIFO 表的底部
LIFO 读	LIFO_RD_DINT LIFO_RD_DWORD LIFO_RD_INT LIFO_RD_UINT LIFO_RD_WORD	把位于 LIFO(后进先出)表的指针存储单元入口数据移走，指针值减 1
LIFO 写	LIFO_WRT_DINT LIFO_WRT_DWORD LIFO_WRT_INT LIFO_WRT_UINT LIFO_WRT_WORD	LIFO 表针增 1，写数据到表里
查找	SEARCH_EQ_BYTE SEARCH_EQ_DINT SEARCH_EQ_DWORD SEARCH_EQ_INT SEARCH_EQ_UINT SEARCH_EQ_WORD	查找所有等于一个给定值的数组值
	SEARCH_GE_BYTE SEARCH_GE_DINT SEARCH_GE_DWORD SEARCH_GE_INT SEARCH_GE_UINT SEARCH_GE_WORD	查找所有大于等于一个指定值的数组值
	SEARCH_GT_BYTE SEARCH_GT_DINT SEARCH_GT_DWORD SEARCH_GT_INT SEARCH_GT_UINT SEARCH_GT_WORD	查找所有比一个指定值大的数组值

功能	*记符*	*描述*
	SEARCH_LE_BYTE SEARCH_LE_DINT SEARCH_LE_DWORD SEARCH_LE_INT SEARCH_LE_UINT SEARCH_LE_WORD	查找所有小于等于一个给定值的数组值
	SEARCH_LT_BYTE SEARCH_LT_DINT SEARCH_LT_DWORD SEARCH_LT_INT SEARCH_LT_UINT SEARCH_LT_WORD	查找所有小于一个给定值的数组值
	SEARCH_NE_BYTE SEARCH_NE_DINT SEARCH_NE_DWORD SEARCH_NE_INT SEARCH_NE_UINT SEARCH_NE_WORD	查找所有不等于一个给定值的数组值
分类	SORT_INT SORT_UINT SORT_WORD	按升序分类一个存储器块
读表	TBL_RD_DINT TBL_RD_DWORD TBL_RD_INT TBL_RD_UINT TBL_RD_WORD	从一个指定表存储单元中复制一个值到输出点
写表	TBL_WRT_DINT TBL_WRT_DWORD TBL_WRT_INT TBL_WRT_UINT TBL_WRT_WORD	从一个输入点中复制一个值到一个指定表存储单元中。

数组传送



当数组传送功能块接收能流，它从一个源存储器块中复制指定数目的元素到目的存储器块中。从输入存储块的变址地址(SR+SNX-1)开始，它复制 N 个元素到输出存储块的变址地址(DS+DNX-1)开始的输出存储块。

注意： 对 ARRAY_MOVE_BOOL,当 16 位寄存器被选中作为源存储器块和目的存储器块中的一的开始地址，指定 16 位寄存器的最低有效位是存储器块的第一位。显示的数值包含 16 位，不管存储器块的长度。

一个数组传送指令的指针是基于 1 的。在使用一个数组传送中，源存储器块或目的存储器块（由它们的开始地址和长度指定）中没有外部元素可以被引用。

The function passes power flow unless one of the following conditions occurs:

功能块传送能流，除非存在下列条件中的一个：

- 接收不到能流
- (N + SNX - 1)大于 Length.
- (N + DNX - 1) 大于 Length

注释： 对每个助记符，对 SR 和 DS 操作数使用相应的数据类型。例如，ARRAY_MOVE_BYTE 要求 SR 和 DS 是字节变量。

数组传送操作数

参量	描述	许用操作数	可选性
Length (??)	每个存储器块的长度（源和目的）；在每个存储器块中元素的数目 1 ≤ Length ≤ 32,767.	常量	No
SR (必须和 DS 是相同的数据类型)	源存储器块的开始地址 **注意：** 对一个布尔类型的数组传送，任何基准地址都能使用；不需要按字节排列。，从指定基准地址开始的 16 位在线显示	除常量外任何操作数。%S - %SC 只在字节、字和双字中允许使用	No
SNX	源存储器块的指针	除在%S - %SC 内变量外任何操作数	No
DNX	目的存储器块的指针	除在%S - %SC 内变量外任何操作数	No
N	计数指示器	除在%S - %SC 内变量外任何操作数	No
DS (必须和 SR 是相同的数据类型)	目的存储器块的开始地址 **注意：**对一个布尔类型的数组传送，任何基准地址都能使用；不需要按字节排列。，从指定基准地址开始的 16 位在线显示	除 S 和常量外任何操作数。%SA- %SC 只在字节、字和双字中允许使用	No

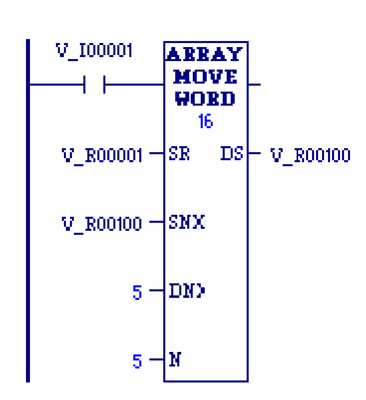
数组传送举例

例 1

定义输入存储块%R0001 - %R0016 和输出存储块%R0100 - %R0115，SQ 设置为%R0001，DS 设置为%R0100,，长度设置为 16。

复制 5 个寄存器%R0003 - %R0007 到寄存器

%R0104 - %R0108，N 设置为 5，SNX=%R0100 设置为 3（指定从%R0001 开始的模块的第三个寄存器%R0003），DNX 设置为 5（指定从%R0100 开始的模块的第五个寄存器%R0104）。



例 2

使用位存储器模块，输入模块从 SR=%M0009 开始，输出模块从%Q0022 开始，两个模块的长度都是 16 个一位寄存器(Length=16)。

为了复制七个寄存器%M0011 - %M0017 到%Q0026 - %Q0032，N 设置为 7，SNX 设置为 3（指定从%M0009 开始的模块的第三个寄存器%M0011），DNX 设置为 5（指定%Q0022 开始的第五个寄存器%Q0026）。

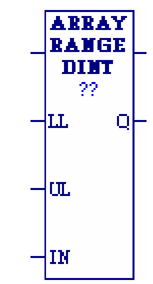
例 3

不按位排序的 16 位从两个在%R00001 (SR)开始的 16 位寄存器传送到%R00100 (DS)开始的两个 16 位寄存器。为了布尔传送的目的，长度设置为 20，因为在任一个存储器块中的其他 12 位都不用考虑。

通过设置 SNX 为 3，N 为 16，DNX 为 5，%R0001 的第三个(SNX)最低有效位到%R0002 的第二个最低有效位（总共有 16 位=N）被写入%R0100 的第五个(DNX)最低有效位到%R0101 的第四个最低有效位（总共 16 位）。

数组范围

ARRAY_RANGE 功能比较一个单精度输入值和两个界定符数组，界定符数组指定上限和下限来决定输入值是否在界定符指定的范围内。输出是一个位组，当输入值大于等于下限并且小于等于上限时被设置为 1。当输入超出限定范围或当限定范围无效时输出为 OFF(0)，如下限超过上限。



助记符:

ARRAY_RANGE_DINT
ARRAY_RANGE_DWORD
ARRAY_RANGE_INT
ARRAY_RANGE_UINT
ARRAY_RANGE_WORD

ARRAY_RANGE 功能比较一个单精度输入值和两个界定符数组，界定符数组指定上限和下限来决定输入值是否在界定符指定的范围内。输出是一个位组，当输入值大于等于下限并且小于等于上限时被设置为 1。当输入超出限定范围或当限定范围无效时输出为 OFF(0)，如下限超过上限。

当 ARRAY_RANGE 接收能量，其将输入参数 IN 中的数值与 LL 和 UL 中的数组元素值指定的各个范围相比较。当 IN 的数值大于等于 LL 的值并且小于等于 UL 的数值时对每个相应的数组元素，输出 Q 设置该位为 ON (1)。当 IN 的数值不在这个范围或该范围无效，即 LL 的值超出 UL 的值，对每个相应的数组元素，输出 Q 设置该位为 OFF(0)。如果操作成功，ARRAY_RANGE 向右传递能流。

数组范围操作数

注意:

- 对每个助记符，对 LL,UL 和 Q 操作数使用相应的数据类型。例如，ARRAY_RANGE_DINT 要求 LL,UL 和 Q 是 DINT 变量。
- Q 不是队列。它以位的格式显示。它为第一个数组元素显示一个 1 (ON) 或 0 (OFF)。对 BOOL 参考地址，Q 代表显示的参考地址。对其它参考地址，Q 代表显示参考地址的最低位。

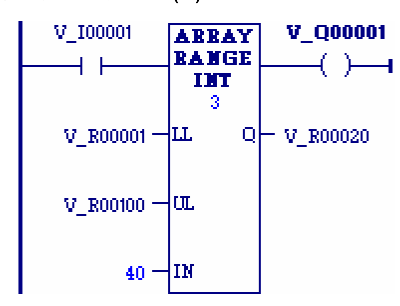
参数	*描述*	*许用操作数*	*可选性*
Length (??)	每个数组中的元素数目	常量	No
LL	范围的下限	除常量和%S - %SC（用于 INT，DINT）外全部操作数	No
UL	范围的上限	除常量和%S - %SC（用于 INT，DINT）外全部操作数	No
IN	与 LL 和 UL 指定的各个范围进行比较的值	.除常量和%S - %SC（用于 INT，DINT）外全部操作数	No
Q	IN 的值在 LL 和 UL 指定的范围内时激活	除 S 外全部操作数	No

数组范围举例

例 1

%R00001 到 %R00008 的下限值是 1, 20, 30, 100, 25, 50, 10 和 200。%R00100 至 %R00108 的上限值是 40, 50, 150, 2, 45, 90, 250 和 47。结果 Q 值将被放在%R00200 的

第一个 8 位。按顺序从低到高的位值是 1, 1, 1, 0, 1, 0, 1 和 0。显示的位值将在%R00200 的低位设置为 ON(1)。OK 输出设置为 ON (1)。



Example 2

下限(LL)数组包含%T00001 至%T00016, %T00017 至%T00032 以及%T00033 至%T00048。下限值是 100、65 和 1。上限值(UL)是 29、165 和 2。结果 Q 的值 0、1、0 将放在%Q00001 至%Q00003。代表%Q00001 值的显示位值将为 0 (OFF)。能量输出设置为 ON (1)。

V_I00001
ARRAY
RANGE
INT
3
V_M00001
V_T00001
LL
Q
V_Q00001
V_T00049
UL
65
IN

FIFO 读



助记符:
FIFO_RD_DINT
FIFO_RD_DWORD
FIFO_RD_INT
FIFO_RD_UINT
FIFO_RD_WORD

先进先出(FIFO) 读功能模块(FIFO_RD)从表中移出数据。数值总是从表的底部移出。如果指针指向最后位值，表就满了。FIFO_RD 必须用来删除指针入口存储单元的记录并且指针减 1。FIFO_RD 和 FIFO_WRT 结合使用，FIFO_WRT 使指针增 1 并且把指针存储单元的内容写入表中。

1. FIFO_RD 复制表的顶部存储单元的值到输出参数 Q 中。然后必须有另外的逻辑程序输入数据到输入存储单元中。
2. 表中剩余的项复制到表中更低的位置。
3. FIFO_RD 将指针减 1。
4. FIFO_RD 执行时，步骤 1，2 和 3 重复执行，直到表为空(PTR = 0)。

当表满时指针首尾不重合。

当 FIFO_RD 接收能量流，表的第一个存储单元的数据被复制到输出 Q。接着，表中的每一项往下移动到下一个更低的存储单元。这从表中的第二项开始，它被移动到存储单元 1。最后，指针减 1。如果最终指针位置变为 0，输出 EM 设为 ON，也就是说，EM 表示表是否为空。

如果指示器大于零并且小于 LEN 指定的数值，FIFO_RD 向右传递能流。

注意：一个 FIFO 表是一个队列。一个 LIFO 表是一个堆栈。

FIFO 读的操作数

注意：对每个助记符，使用跟 TB 和 Q 操作数相应的数据类型。例如，FIFO_RD_DINT 要求 TB 和 Q 是 DINT 变量

.

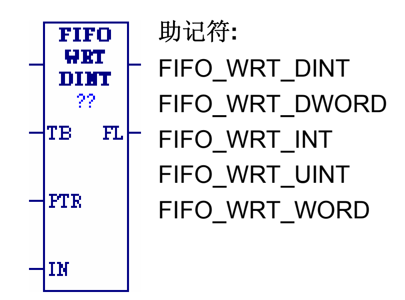
参数	*描述*	*许用操作数*	*可选性*
Length (??)	1 ≤ Length ≤ 32,767.	常量	No
TB (必须和 Q 是相同的类型)	FIFO 表中的元素	除常量外任何操作数	No
PTR	指针。FIFO 表最后的元素的指针	除常量，数据流以及%S -%SC 变量外的任何操作数	No
EM	当读表中最后元素时激活	流	No
Q（必须与 TB 类型相同）	从 FIFO 表中读的元素	除常量外的任何操作数，S; SA, SB, SC 只用于字和双字	No

FIFO 读举例

PRODUCT 是一个具有 100 个字元素的 FIFO 表。当使能输入端 PACK_IT 是 ON 时，表中存储单元由 STK_PTR 指向的 PRODUCT 数据项被复制到 CART 指定的参考地址。指向的表位置将在底部或是表中最先进入的数据项。STK_PTR 中的数字减 1。因为在连续的 PACK_IT 触发期间当前数据被复制到外部，在 PRODUCT 表中的最先进入的数据项的副本被留在每个表中存储单元的后面。当 PTR=0 时，输出点 EM 传递能流，触发线圈 EMPTY。表 PRODUCT 中没有数据能够在未使用 FIFO_WRT 功能块先进行数据复制而被读取。

PACK_IT
FIFO
RD
WORD
100
EMPTY
PRODUCT
TB
EM
STK_PTR
PTR
Q
CART

FIFO 写



助记符:
FIFO_WRT_DINT
FIFO_WRT_DWORD
FIFO_WRT_INT
FIFO_WRT_UINT
FIFO_WRT_WORD

先进先出(FIFO)写(FIFO_WRT) 功能块传送数据到表中。功能块使表指针的值加 1，在 FIFO 表中的新的指针位置增加一个入口。数值总是被传进到表的底部。如果指针指向了最后的位置，并且表已经满了，FIFO_WRT 就不能增加任何数值了。FIFO_RD 功能块必须用来移除指针存储单元的记录，并使指针减 1。

1. FIFO_WRT 使指针值加 1。
2. FIFO_WRT 从输入参数 IN 复制数据到指针指向的表中的位置。（它改写当前在该位置的任何数值）。必须使用另外的逻辑程序把数据放到输入单元中。
3. FIFO_WRT 执行时每次都重复步骤 1 和 2，直到表满为止(PTR=0)。

当表满时指针首尾不重合

当 FIFO_WRT 接收能量时，指针值减 1。接着，输入数据被写入表中指针指示位置。如果指针已经指向表中最后的位置，就写入任何数据了，FIFO_WRT 也不能向右传递能流。指针总是指向输入表中的最后一个数据项。如果表已经满了，就不可能向表中增加任何输入了。

FIFO_WRT 成功执行后，向右传递能流 (PTR < LEN)。

FIFO 写的操作数

注意：对每个助记符，使用跟 TB 和 IN 操作数相应的数据类型。例如，FIFO_WRT_DINT 要求 TB 和 IN 是 DINT 变量。

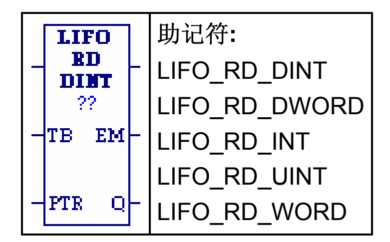
参数	*描述*	*许用操作数*	*可选性*
Length (??)	1 ≤ Length ≤ 32,767.	常量	No
TB (必须和 IN 是相同的数据类型)	FIFO 表中的元素	除常量、数据流和 S 外任何操作数。S A– SC 只允许在字和双字类型中使用	No
PTR	指针。FIFO 表最后一个元素的指针	除常量、数据流和 S – SC 外任何操作数	No
IN (必须和 TB 是相同的数据类型)	写入 FIFO 表的元素	All. S – SC allowed only for WORD, DWORD types. 任何操作数。S – SC 只允许在字和双字类型中使用	No
FL	当 IN 写入表中最后一个位置时激活	能流	No

]

FIFO 写举例

PRODUCT 是一个具有 100 个字元素的 FIFO 表。当使能输入端 UNPACK 打开时，P_CODE 中的一个数据项被复制到 STK_PTR 指向的表中位置。当 PTR=LEN 时，输出点 FL 传送能量，激活 FULL 线圈。P_CODE 中没有数据能够在未使用 FIFO_RD 功能块先把数据复制出来而写入表中。

LIFO 读



后进先出（LIFO）读(LIFO_RD)功能块把数据从表中移出。数值总是从表的顶部移走。如果指针到达了最后位置并且表已经满了，FIFO_RD 功能块必须用来移除指针存储单元的记录，并使指针减 1。LIFO_RD 和 LIFO_WRT 结合使用，LIFO_WRT 使指针减 1 并写入数据到表中。

1. LIFO_RD 复制指针指示的数据到输出参数 Q 中。然后必须有另外的逻辑程序输入数据到输入单元中。
2. LIFO_RD 将指针减 1。
3. FIFO_RD 执行时，步骤 1，2 和 3 重复执行，直到表为空(PTR = LEN)。

当表满时指针首尾不重合

当 LIFO_RD 接收能量流，指针位置中的数据被复制到输出 Q 中，接着指针减 1。当指针位置变为 0，输出 EM 设置为 ON，也就是说，EM 表示表是否为空。当 LIFO_RD 接收能量流时，如果表是空的，不能读取数据。指针总是指示输入表中的最后数据项。

如果指针在要读的元素的范围之内，LIFO_RD 向右传递能流。

注意：一个 LIFO 表是一个堆栈。一个 FIFO 表是一个队列。

LIFO 读操作数

注意： 对每个助记符， TB 和 Q 操作数要使用相应的数据类型。例如，LIFO_RD_DINT 要求 TB 和 Q 是 DINT 变量。

.

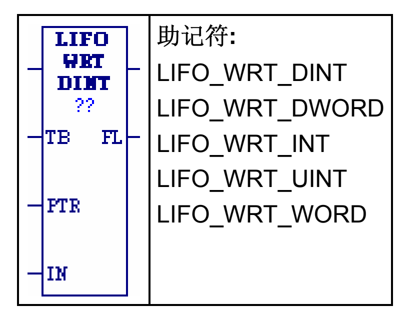
参数	*描述*	*许用操作数*	*可选性*
Length (??)	1 ≤ Length ≤ 32,767.	常量	No
TB (必须和 Q 是相同的类型)	表中的元素	除常量外任何操作数	No
PTR	指针。接着要读元素的指针	除常量，数据流以及%S -%SC 变量外的任何操作数	No
EM	当读表中最后元素时激活	流	No
Q（必须与 TB 类型相同）	从 FIFO 表中读的元素	除常量外的任何操作数，S; SA, SB, SC 只用于字和双字	No

LIFO 读举例

PRODUCT 是一个具有 100 个字元素的 LIFO 表。当使能输入端 PACK_IT 打开时，表的顶端的数据项被复制到 CART 指定的参考地址。STK_PTR 确定的参考地址包含表指针。输出线圈 EMPTY 激活表示表是空的。

PACK_IT
LIFO
RD
WORD
100
EMPTY
PRODUCT
TB
EM
STK_PTR
PTR
Q
CART

LIFO 写



助记符:
LIFO_WRT_DINT
LIFO_WRT_DWORD
LIFO_WRT_INT
LIFO_WRT_UINT
LIFO_WRT_WORD

后进先出（LIFO）写(LIFO_WRT)功能块使表指示器的值加 1，在表中的新的指针存储单元增加一个入口。数值总是被传进到表的顶部。如果指针指向了最后的位置，并且表已经满了，LIFO_WRT 就不能增加任何数值了。LIFO_WRT 功能块必须用来移除指针存储单元的记录，并使指针减 1

1. LIFO_WRT 将表指针加 1。
2. LIFO_WRT 从输入参数 IN 中复制数据到指针指示的表中的位置。（它改写当前在那个位置的任何数值。）然后必须有另外的逻辑程序输入数据到输入单元中。
3. 每次 LIFO_WRT 执行时步骤 1 和 2 重复，直到表满为止(PTR=LEN)。

当表满时指针首尾不重合

当 LIFO_WRT 接收能量流，指针加 1，接着新的数据写入指针存储单元。如果指针已经在表中的最后位置，没有数据能够写入，并且 LIFO_WRT 不能向右传送能流。指针总是指示输入表中的最后数据项。如果表是满的，就不可能增加更多的数据。

LIFO_WRT 成功执行之后向右传送能流(PTR < LEN)

注意: LIFO 是一个堆栈。FIFO 是一个队列。

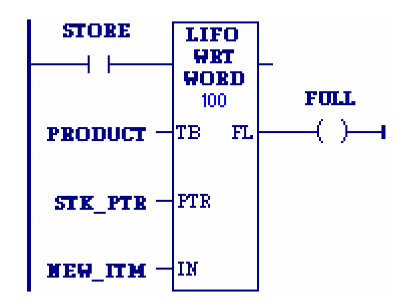
LIFO 写操作数

注意：对每个助记符，TB 和 IN 操作数要使用相应的数据类型。例如，LIFO_ WRT _DINT 要求 TB 和 IN 是 DINT 变量。

参数	描述	许用操作数	可选性
Length (??)	1 ≤ Length ≤ 32,767.	常量	No
TB (必须和 IN 是相同的数据类型)	表中的元素	除常量、数据流和 S 外任何操作数。S A– SC 只允许在字和双字类型中使用	No
PTR	指针。下一个写进元素的指针	除常量、数据流和 S – SC 外任何操作数	No
IN (必须和 TB 是相同的数据类型)	写入表的元素	任何操作数。S – SC 只允许在字和双字类型中使用	No
FL	当 IN 写入表中最后一个位置时激活	能流	No

LIFO 写举例

PRODUCT 是一个 100 字元素的 LIFO 表。当使能输入端 STORE 打开时，NEW_ITEM 中的一个数据项被复制到 STK_PTR 中的数值指示的表中存储单元。当 PTR = LEN 时，输出 FL 传送能量，激活 FULL 线圈。没有使用 LIFO_RD 功能块把数据复制出去之前，NEW_ITEM 中没有数据能够增加到表中，



查找

当查找功能块使能激活，其在指定的存储块中查找满足查找标准的数值。例如，SEARCH_GE_DWORD 查找一个大于等于指定数值（IN 操作数）的双字。

查找能对六种数据类型的六种不同的关系进行操作，总共有 36 个助记符。

查找关系：

SEARCH_EQ_...	查找等于 IN 操作数的指定数据类型的数值
SEARCH_GE_...	查找大于等于 IN 的指定数据类型的数值
SEARCH_GT_...	查找大于 IN 的指定数据类型的数值
SEARCH_LE_...	查找小于等于 IN 的指定数据类型的数值
SEARCH_LT_...	查找小于 IN 的指定数据类型的数值
SEARCH_NE_...	查找不等于 IN 的指定数据类型的数值

数据类型

BYTE, DINT, DWORD, INT, UINT, WORD

在 AR+INX 开始的查找，其中 AR 是开始地址，INX 是指向存储块的变址值。查找持续到满足查找条件的寄存器被找到或存储块被查找结束。

- 如果一个寄存器找到，Found 指示（FD）设置为 ON，输出变址(ONX)设置为指向该块中寄存器的相对位置。
- 如果存储块被查找结束之前没有寄存器被找到，Found 指示（FD）设置为 OFF，输出变址(ONX)设为 0。

输入变址(INX)是基于 0 的，即 0 是第一个基准，而输出变址(ONX)是基于 1 的，即 1 是第一个基准。

INX 的有效值是 0 到(Length - 1)。ONX 的有效值是 1 到 Length 。

INX 应该设置为 0 开始在存储块中的第一个寄存器的查找。这个值执行一次就加 1。如果输入 INX 的数值超出了范围(< 0 或 > Length-1)，INX 设置为默认值 0。

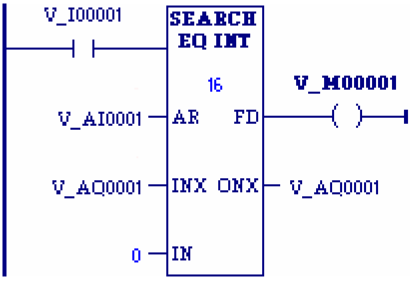
SEARCH 执行无误时向右传送能流。如果 INX 超出范围，SEARCH 就不向右传送能流。

查找功能块操作数

注意：对每个助记符，AR 和 IN 操作数要使用相应的数据类型。例如，
SEARCH_EQ_BYTE 要求 AR 和 IN 是字节变量。

参数	*描述*	*许用操作数*	*可选性*
Length (??)	从构成查找存储器块的 AR 开始的寄存器数目。1 ≤ Length ≤ 32,767，8 位或 16 位寄存器。	常量	No
AR（必须与 IN 具有相同的类型 ）	查找的存储块的开始地址；在存储块中的第一个寄存器地址。	除常量外任何操作数	No
INX	指向开始查找的存储器块的基于 0 的变址。O 指向第一个参考。有效范围：0 ≤ INX ≤ (Length-1)。如果 INX 超出范围，它被设置为默认值 0。	除常量外任何操作数	No
IN（必须与 AR 具有相同的类型）	查找数值的依据。例如： SEARCH_GT_DINT 查找一个大于 IN 的 DINT 数值。 SEARCH_NE_UINT 查找一个不等于 IN 的 UINT 数值。 SEARCH_GE_WORD 查找一个大于等于 IN 的 WORD 数值。	任何操作数	No
ONX	查找目标所在的存储块内的基于 1 的位置。1 指向第一个给定值。数值范围：1 ≤ ONX ≤ Length	数据流，I, Q, M, T, G, R, P, L, AI, AQ	No
FD	找到指示。当一个满足查找标准的寄存器被找到并且功能模块成功执行时，这个能量流指示器激活。	能量流	No

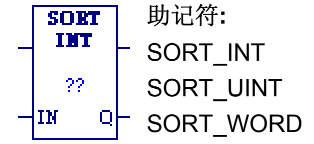
查找功能举例



为了查找存储块%AI00001 - %AI00016，AR 设为%AI00001,长度设为 16。16 位寄存器的数值是 100, 20, 0, 5, 90, 200, 0, 79, 102, 80, 24, 34, 987, 8, 0 和 500 。最初，指向 AR, %AQ0001 的查找指针是 5。当能流输入为 ON，每次扫描在存储器块中查找与 IN 数值 0 相匹配数。第一次扫描开始在%AI00006 查找，并在%AI00007 找到一个匹配值，所以 FD 指示为 ON，%AQ00001 变为 7。第二次扫描从%AI00008 开始，并在%AI00015 找到一个匹配值，所以 FD 仍然指示为 ON，%AQ00001 变为 15。下一次扫描从%AI000016 开始。查找到存储器尾部而没有找到一个匹配值时，FD 设为 OFF，%AQ0001 设为 0。下一次扫描从存储块起始开始查找。

分类

SORT 功能块接收能量流，按升序分类存储块 IN 的元素。输出存储器块 Q 包含指示在原存储器块或表中的分类元素指针的整数。Q 与 IN 的大小完全相同。Q 也有一个要分类的元素的数目的规定（LEN）



助记符:
SORT_INT
SORT_UINT
SORT_WORD

SORT 对不多于 64 个元素的存储器块操作。当 EN 端为 ON，IN 中所有的元素根据的数据类型按升序分类。数组 Q 也被创建，用于表明在无序数组中每个分类元素的原始位置。OK 总是设置为 ON。

注意:　SORT 在每次扫描到有使能激活时执行。.

不要在一个定时或触发输入程序块中使用 SORT 功能块。

操作数

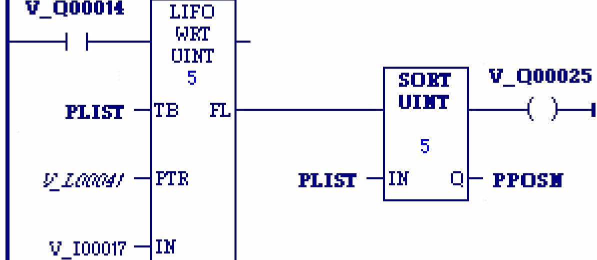
注意:　对每个助记符，IN 和 Q 操作数要使用相应的数据类型。例如，SORT_INT 要求 IN 和 Q 是 INT 变量。

参数	*描述*	*许用操作数*	*可选性*
Length (??)	构成要分类存储器块的元素的数目(1—64)	常量	No
IN	含有要分类的元素的存储器块。分类之后，IN 包含在分类次序中的元素。	除数据流，S，常量外任何操作数。SA – SC 只在字类型中有效	No
Q（必须与 IN 具有相同的类型）	给出在原始存储器块中的已经分类元素的位置的一个下标数组	除 S – SC 和常量外何操作数	No

范例

每次%Q00014 为 ON 时，新的部分数(%I00017 - %I00032)被推进一个部分数组 PLIST。当数组被填满时，它被分类并且输出%Q00025 显示为 on。于是，数组 PPOSN 包含在 PLIST 中分类已经完成之前的正在分类元素占用的原始地址

如果 PLIST 是一个有 5 个元素的数组，排序前包括数值 25, 67, 12, 35, 14，排序后它就包括数值 12, 14, 25, 35, 67。PPOSN 就包含数值 3, 5, 1, 4, 2。



读表

读表(TBL_RD)功能块连续地在一个表中读数据。当指针到达表尾，它重新指向表头。(TBL_RD 正如 FIFO_RD 一样循环)



助记符:
TBL_RD_DINT
TBL_RD_DWORD
TBL_RD_INT
TBL_RD_UINT
TBL_RD_WORD

当 TBL_RD 接收能流时：

1. TBL_RD 使指针加 1。
2. TBL_RD 将指针指定的数据复制到输出参数 Q 中。接着必须有另外的逻辑程序从输出基准地址中获取数据。
3. 每次指令执行时，重复第一步和第二步直到表的末端（PTR=在 Length 中指定的长度）。当到达表的末端时，指针回到表的开始。

当 TBL_RD 接收能流，指针（PTR）加 1。如果新的指针位置是表中最后一项，输出 EM 设置为 ON 。下一次 TBL_RD 执行时，PTR 自动置回 1。PTR 增加之后，在新的指针位置的内容被复制到输出 Q 中。

当 TBL_RD 有使能输入，就相右传递能流。

注意：TBL_RD 和 TBL_WRT 功能块能在相同或不同的表中执行。通过给指针指定一个不同的参考地址，这些功能块能在不同的位置或以不同的速率访问相同的数据表。

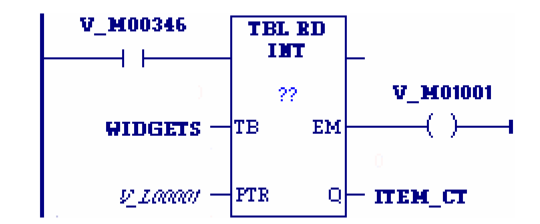
操作数

注意： 对每个助记符，TB 和 Q 操作数要使用相应的数据类型。例如，TBL_RD_DINT 要求 TB 和 Q 是 DINT 变量。

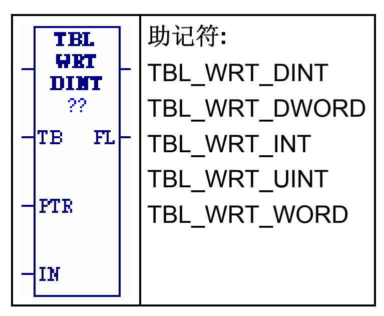
参数	*描述*	*许用操作数*	*可选性*
Length	1 ≤ Length ≤ 32,767	常量	No
TB（必须与 Q 类型相同）	表中的元素	除常量外任何操作数	No
PTR	指针。下一个元素的指针	除数据流、S – SC、常量外任何操作数	No
EM	读表中最后一个元素时激活	能流	No
Q （必须与 TB 类型相同）	从表中读的元素	除常量、S 外任何操作数。SA, SB, SC 只在字和双字中允许	No

范例

WIDGETS 是一个具有 20 个整型元素的表。当使能输入端%M00346 打开时，指针增加，表中下一个元素的内容被复制到 ITEM_CT。%L00001 功能块作为指向数据表的的指针。%M01001 被用来指示数据表中所有的项都已经被访问。



写表



写表(TBL_WRT)功能块连续更新从不变满的一个表中的数值。当指针（PTR）到达表尾，其自动返回到表的顶端。

1. TBL_WRT 使指针值加 1。
2. TBL_WRT 从输入参数 IN 中复制数据到指针指定的表中的位置。（改写当前在那个位置的任何数值）接着必须有另外的逻辑程序把数据放置到输入基准中。
3. 指令执行时步骤 1 和 2 每次都重复进行，直到表满为止(PTR=LEN)。

当表满时，指针回到表头。

注意： TBL_WRT 和 TBL_RD 功能块能对相同或不同的表操作。通过为指针指定一个不同的参考地址，这些功能块能够在不同的位置或以不同的速率访问相同的数据表。

当 TBL_WRT 接收能流时，指针(PTR)加 1。如果这个新的指针位置是表中的最后一项，输出 FL 设为 ON。下一次 TBL_WRT 执行时，PTR 自动返回到 1。PTR 值增加之后，TBL_WRT 把输入参考地址的内容写入当前指针存储单元，改写已经存储在那的数据。

TBL_WRT 有使能输入，就相右传递能流。

注意：TBL_WRT 象 FIFO_WRT 一样指针首尾不重合。

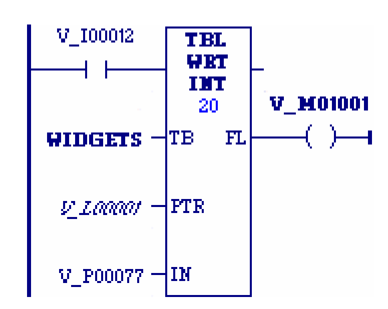
操作数

注意： 对每个助记符，TB 和 IN 操作数要使用相应的数据类型。例如，TBL_WRT_DINT 要求 TB 和 IN 是 DINT 变量。

参量	*描述*	*允许的*	*可选性*
Length	1 ≤ Length ≤ 32,767.	常量	No
TB（必须与 IN 类型相同）	表中的元素	除 S、常量、数据流外任何操作数。SA – SC 只在字和双字中允许	No
PTR	指针。下一个元素指针	除常量、数据流、%S - %SC 外任何操作数	No 否
IN（必须与 TB 类型相同）	要写入表中的元素	任何操作数。%S - %SC 只允许在字和双字中使用	No
FL	IN 被写入表中最后一个元素中时激活	能流	No

写表举例

WIDGETS 是一个有 20 个整型元素的表。当使能输入端%I00012 是 ON，指针增加，%P00077 的内容被写入表中指针指向的存储单元。%L00001 功能块作为进入数据表的指针。



数学功能

在使用一个数学或数字功能之前，编制的程序可能需要包含转换数据类型的逻辑。每个功能块的描述包含适当的数据类型的信息。在8-错误！未定义书签。的“转换功能块”部分解释了如何把数据转换为一个不同的类型。

功能	助记符	描述
绝对值	ABS_DINT ABS_INT ABS_REAL	求一个双精度整数、单精度整数或浮点数的绝对值。助记符指定了数值的数据类型。
加	ADD_DINT ADD_INT ADD_REAL ADD_UINT	加法。将两个数相加。
除*	DIV_DINT DIV_INT DIV_MIXED DIV_REAL DIV_UINT	除法。一个数除于另一个数，输出商。 **注意：** 执行除法操作时注意避免溢出情况发生
模数	MOD_DINT MOD_INT MOD_UINT	除法求模。一个数除于另一个数，输出余数。
乘*	MUL_DINT MUL_INT MUL_MIXED MUL_REAL MUL_UINT	.乘法。两个数相乘。 **注意：** 执行乘法操作时注意避免溢出
比例	SCALE	把一个输入参数比例放大或缩小，把结果放在输出单元
减	SUB_DINT SUB_INT SUB_REAL SUB_UINT	减法。从另一个数中减去一个

- 当乘或除 16 位数时避免溢出，使用在 8－60 描述的转换功能把该数转换成 32 位格式

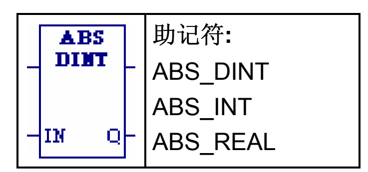
当一个操作结果溢出时，就没有能流。如果对一个 INT 或 DINT 操作数的操作导致溢出，输出参考设置为该数据类型的最大可能值。对有符号数，符号被设置为指示溢出的方向。如果有符号数或双精度整数被使用，除法和乘法功能块的结果的符号取决于 I1 和 I2 的符号。

如果对一个 UNIT 操作数的操作导致溢出，结果设置为最小值（0）。如果对一个 UNIT 的操作导致溢出，结果设置为最大值。

如果操作没有导致溢出，能流输出打开，除非下列无效浮点数操作之一发生。在这些情况下，能流设为 OFF.

- 加法(+ ∞); 减法, (± ∞)-(± ∞)
- 乘法, 0 x ∞
- 除法， 0 除 0.
- 除法，∞ 除∞
- I1 和/或 I2 不是一个数值

绝对值



当功能块接收能量流，其将输入 IN 的绝对值送至输出 Q。

功能块输出能流，除非下列情况之一发生：

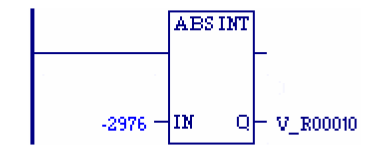
- 对于 INT 类型，IN 是 MININT
- 对于 DINT 类型，IN 是 MINDINT
- 对于 REAL 类型，IN 不是数值

操作数

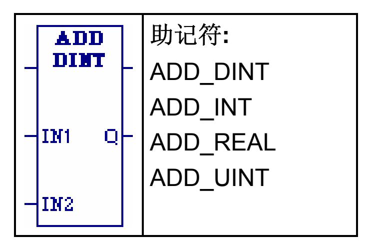
参量	*描述*	*许用操作数*	*可选性*
IN（必须与 Q 类型相同）	处理的数值	除 S, SA, SB, SC 外任何操作数	No
Q（必须与 IN 类型相同）	IN 的绝对值	除 S, SA, SB, SC 和常量外任何操作数	No

范例

将－2，976 的绝对值 2，976 输出到%R00010：



加



助记符:
ADD_DINT
ADD_INT
ADD_REAL
ADD_UINT

当 ADD 功能块接收能流时，其将具有相同数据类型的两个操作数 IN1 和 IN2 相加并将总和存储在赋给 Q 的输出变量中。

当 ADD 执行无溢出时，能流输出激活，除非发生一个无效操作。如果一个 ADD_DINT, ADD_INT 或 ADD_REAL 操作导致溢出，Q 设置为具有适当符号的最大可能值并且没有能流。如果 ADD_UINT 操作导致溢出，Q 设置为最小值。

助记符	*操作*	*显示*
ADD_INT	Q(16 bit) = IN1(16 bit) + IN2(16 bit)	带符号十进制数，5 位数
ADD_DINT	Q(32 bit) = IN1(32 bit) + IN2(32 bit)	带符号十进制数，10 位数
ADD_REAL	Q(32 bit) = IN1(32 bit) + IN2(32 bit)	十进制数，带符号和小数，8 位数（包括小数位）
ADD_UINT	Q(16 bit) = IN1(16 bit) + IN2(16 bit)	无符号十进制数，5 位数

ADD 功能块操作数

操作数	描述	允许操作数	可选性
IN1	等式 IN1+IN2=Q 加号左边的数值	除 S, SA, SB, SC 外任何操作数	No
IN2	等式 IN1+IN2=Q 加号右边的数值	除 S, SA, SB, SC 外任何操作数	No
Q	IN1+IN2 的结果。如果发生溢出，结果是最大的可能值并且无能流	除 S, SA, SB, SC 和常量外任何操作数	No

ADD 举例

第一个例子试图建立一个能计算开关%I0001 闭合次数的计算回路，但不成功。运行结果存储在寄存器%R0002 中。这个设计的目的是当%I0001 闭合时，ADD 指令将%R0002 中的数值加 1，并将新的数值返回到%R0002。这个设计的问题是%I0001 闭合时，ADD 指令执行一次时间为一个 PLC 扫描时间。所以，例如，%I0001 保持闭合状态 5 次扫描时间，输出就将增加 5 次，即使%I0001 在那个时期只闭合了一次。

为了解决上述问题，ADD 指令的使能输入应该来自一个跳变（单触发）线圈，如下面所述。在改良的电路里，%I0001 输入开关控制一个跳变（单触发）线圈，%M0001 的触点接通 ADD 功能块的使能输入，每次扫描%M000 使触点%I0001 闭合一次。为了使%M0001 触点再次闭合，触点%I0001 只能再次分来闭合。

注意：如果 IN1 和或 IN2 是非数，ADD_REAL 不传送能流。

Divide

当 DIV 功能块接收能量流，操作数 IN1 被与 IN1 具有相同数据类型的操作数 IN2 除并且将商存储在输出变量 Q 中，商的数据类型也与 IN1 和 IN2 相同。

当 DIV 执行无溢出时，能流输出激活，除非发生无效操作。如果一个溢出发生，结果是带适当符号的最大可能值，能流断开。



助记符:
DIV_DINT
DIV_INT
DIV_MIXED
DIV_REAL
DIV_UINT

注意：

- DIV 直接舍掉小数部分，不是取最接近的整数商。例如，24/5=4。
- DIV_MIXED 使用混合数据类型。
- 注意避免溢出。

助记符	*操作数*	*显示*
DIV_UINT	Q(16 bit) = IN1(16 bit) / IN2(16 bit)	无符号十进制数，5 位数
DIV_INT	Q(16 bit) = IN1(16 bit) / IN2(16 bit)	带符号十进制，5 位
DIV_DINT	Q(32 bit) = IN1(32 bit) / IN2(32 bit)	带符号十进制，10 位数
DIV_MIXED	Q(16 bit) = IN1(32 bit) / IN2(16 bit)	带符号十进制，5 位数
DIV_REAL	Q(32 bit) = IN1(32 bit) / IN2(32 bit)	十进制数，带符号和小数，8 位数（包括小数位）

DIV_UINT, DIV_INT, DIV_DINT, and DIV_REAL 的操作数

DIV_UINT, DIV_INT, DIV_DINT, 和 DIV_REAL 操作数

参量	*描述*	*许用操作数*	*可选性*
IN1	被除数，等式 IN1/IN2＝Q 除号左边的数	除 S, SA, SB, SC 外任何操作数	No
IN2	除数，等式 IN1/IN2＝Q 除号右边的数	除 S, SA, SB, SC 外任何操作数	No
Q	IN1/IN2 的商。如果发生溢出，结果是带适当符号的最大值并能流断开。	除 S, SA, SB, SC 和常量外任何操作数	No

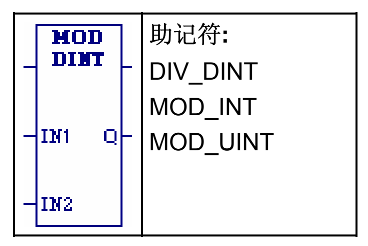
DIV_MIXED 操作数

参量	*描述*	*许用操作数*	*可选性*
IN1	被除数，等式 IN1/IN2＝Q 除号左边的数	除 S, SA, SB, SC 外任何操作数	No
IN2	被除数，等式 IN1/IN2＝Q 除号右边的数	除 S, SA, SB, SC 外任何操作数	No
Q	IN1/IN2 的商。如果发生溢出，结果是带适当符号的最大值并能流断开。	除 S, SA, SB, SC 和常量外任何操作数	No

范例

DIV_DINT 能与 MUL_DINT 功能块一起使用来缩放一个±10V 输入到±25,000 工程单位。见 8-124“举例－缩放一个模拟输入”

模



当除法求模(MOD)功能块接收能流，输入 IN1 除以 IN2 并输出余数到 Q。

这三个操作数都必须是相同的数据类型。结果的符号总是与输入参数 IN1 的符号相同。输出 Q 使用下面的公式计算得出：

Q = IN1-((IN1 DIV IN2) * IN2)

式中 DIV 得出一个整数。

当功能块接收能量流时，能流输出总是 ON，除非有除数是 0。在那种情况下，能量流输出设为 OFF。

Modulus 功能块操作数

参量	描述	许用操作数	可选性
IN1	获得余数的被除数。等式 IN1 MOD N2=Q 中“MOD”左边的数	除 S, SA, SB, SC 外任何操作数	No
IN2	除数。等式 IN1％MOD IN2=Q 中“MOD”右边的数	除 S, SA, SB, SC 外任何操作数	No
Q	IN1/IN2 余数	除 S, SA, SB, SC 和常量任何操作数	No

乘法



助记符:
MUL_DINT
MUL_INT
MUL_MIXED
MUL_REAL
MUL_UINT

当 MUL 功能块接收能量流，操作数 IN1 乘于与 IN1 具有相同数据类型的操作数 IN2 除并且将结果存储在输出变量 Q 中。

当 MUL 执行无溢出时，能流输出激活，除非发生无效操作。如果一个溢出发生，结果是带适当符号的最大可能值，能流断开。(对于 MUL_UINT，溢出将得到 0)

注意：MUL_MIXED 使用混合数据类型。注意避免溢出。

助记符	操作	显示
MUL_INT	Q(16 bit) = IN1(16 bit) * IN2(16 bit)	带符号十进制数，5 位
MUL_DINT	Q(32 bit) = IN1(32 bit) * IN2(32 bit)	带符号十进制数，10 位
MUL_REAL	Q(32 bit) = IN1(32 bit) * IN2(32 bit)	十进制数，带符号和小数，8 位（包括小数位）
MUL_UINT	Q(16 bit) = IN1(16 bit) * IN2(16 bit)	无符号十进制数，5 位
MUL_MIXED	Q(32 bit) = IN1(16 bit) * IN2(16 bit)	带符号十进制数，10 位

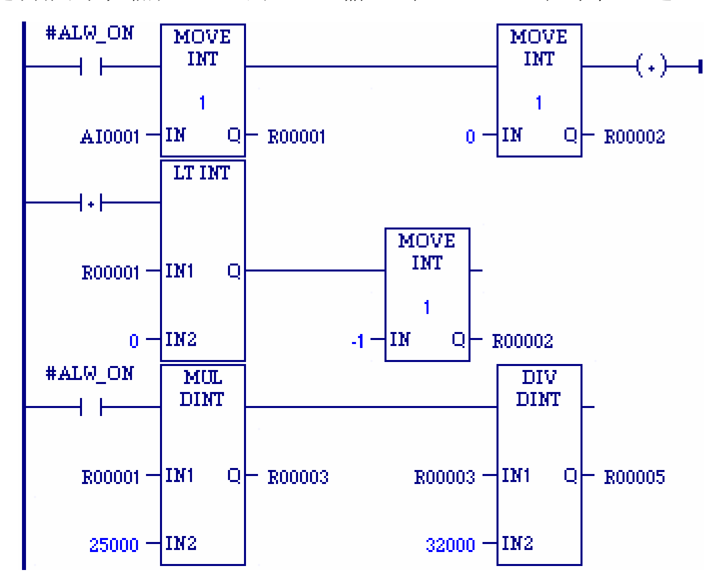
乘法操作数

参量	描述	许用操作数	可选性
IN1	乘式第一个数，等式 IN1 * IN2=Q 中乘号左边的数	除 S, SA, SB, SC 外任何操作	No
IN2	乘式第二个数，等式 IN1 * IN2=Q 中乘号右边的数	除 S, SA, SB, SC 外任何操作	No
Q	IN1*IN2 的结果。如果发生溢出，结果是带适当符号的最大值，能流输出断开。	除 S, SA, SB, SC 和常量外任何操作	No

举例-缩放模拟输入值

一个通用应用软件是通过一个乘法操作后接着一个除法操作，可能还有加法运算来缩放模拟输入值。一个 0 到 ±10V 的模拟输入将替代在它相应%AI 输入寄存器中的值 0 至 ±32,000。使用一个 MUL_INT 功能块乘以%AI 输入寄存器的值将产生一个溢出，因为一个 INT 类型指令输入输出范围在 32,767 至–32,768。使用%AI 的值作为 MUL_DINT 的输入值也不能运算，因为 32 位的 IN1 将同时合并 2 个模拟量输入。为了解决这个问题，你可以把模拟输入移至一个双寄存器的低字，接着检验符号，如果符号检验为正的则设置第二个寄存器为 0，检验为符号则设置为－1。然后使用由一个 MUL_DINT 建立的双寄存器，MUL_DINT 给出一个 32 位结果，也可以和后面的 DIV_DINT 功能块一起使用。

例如，下面的逻辑能用来缩放%AI1 的±10V 输入到±25000 工程单位送进%R5。



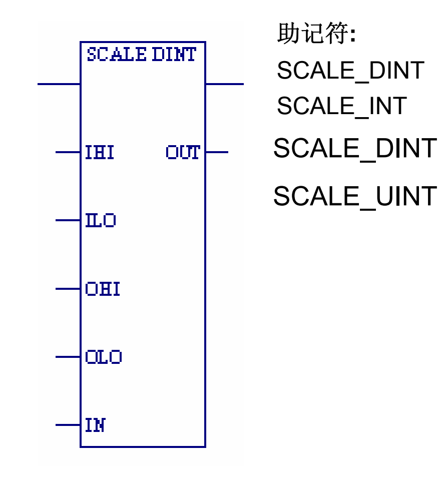
一个备用的，但不那么精确的，使用 INT 数值设计这个电路的方法是先放好 DIV_DINT 指令，后面接着 MUL_DINT 指令。DIV 指令中 IN2 的值为 32，MUL 中 IN2 的值为 25。

经过这样处理，将保持上面电路的缩放比例，并保持数值在 INT 类型指令的工作范围之内。然而，DIV 指令具有丢弃余数的固有特性，所以当 DIV 输出乘以 MUL 指令时，也要乘上因为丢弃的余数引入的误差。误差的百分比在输入值的整个范围是非线性的，输入小，误差的百分比大。

通过对比，在上面的例子中，由于 DIV 操作最后执行，结果更精确，所以丢弃的余数也没有被乘进去。如果要求更高的精度，在这个例子中用 REAL 类型数学指令代替使余数不被丢弃。

缩放

当缩放功能块接收能量流，它缩放输入操作数 IN 并把结果放到由输出操作数 OUT 指派的输出变量中。当 SCALE 操作无溢出时，能流输出激活。



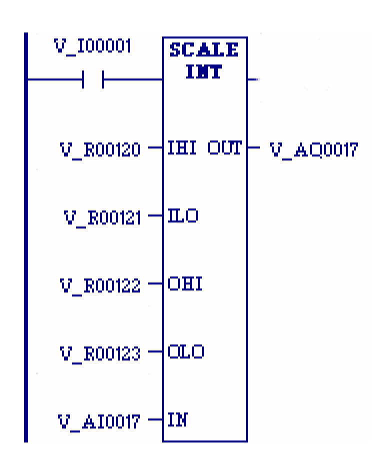
助记符:
SCALE_DINT
SCALE_INT
SCALE_DINT
SCALE_UINT

操作数

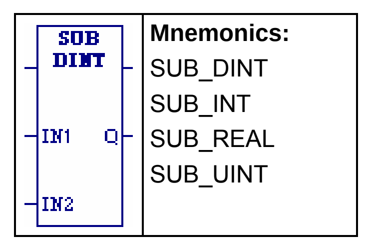
参量	描述	许用操作数	可选性
IHI	(输入上限)最大输入值（与模块有关）。未缩放数的上限。IHI 是与 ILO,OHI 和 OLO 一起使用来计算应用于输入 IN 的缩放因子。	除 S, SA, SB, SC 外任何操作数	No
ILO	(输入下限)最小输入值（与模块有关）。未缩放数的下限。必须与 IHI 数据类型相同。	除 S, SA, SB, SC 外任何操作数	No
OHI	（输出上限）最大输出值。缩放数据的上限。必须与 IHI 数据类型相同。当 IN 输入是 IHI 数值，输出 OUT 值必须与 OHI 数值相同。	除 S, SA, SB, SC 外任何操作数	No
OLO	（输出下限）最小输出值。缩放数据的下限。必须与 IHI 数据类型相同。当 IN 输入是 ILO 数值，输出 OUT 值必须与 OLO 数值相同。	除 S, SA, SB, SC 外任何操作数	No
IN	（输入值）被缩放的数值。必须与 IHI 数据类型相同。	除 S, SA, SB, SC 外任何操作数	No
OUT	（输出值）输入值经缩放后的等效值。必须与 IHI 数据类型相同。	除 S, SA, SB, SC 外任何操作数	No

范例

在这个例子中，寄存器%R0120 到%R0123 都用来存储缩放数值的高低限。要被缩放的输入数是模拟量输入%AI0017。缩放输出数据被用来控制模拟输出%AQ0017。%I0001 是 ON，执行缩放操作。



减法



当 SUB 功能块接收能流，操作数 IN1 减去与 IN1 具有相同数据类型的操作数 IN2 除并且将结果存储在具有相同数据类型的由 Q 指定的输出变量中。

当 SUB 执行无溢出时，能流输出激活，除非发生无效操作。对 SUB_INT, SUB_DINT, and SUB_REAL,如果一个溢出发生，结果是带适当符号的最大可能值，能流输出断开。

If a SUB_UINT operation results in a negative number, Q wraps around. (For example, a result of –1 sets Q to 65535.)

如果一个 SUB_UINT 操作得到一个负数，Q 返回（例如，结果为－1 则 Q 输出为 65535）。

助记符	操作	显示
SUB_INT	Q(16 bit) = IN1(16 bit) – IN2(16 bit)	带符号十进制数，5 位
SUB_DINT	Q(32 bit) = IN1(32 bit) – IN2(32 bit)	带符号十进制数，10 位
SUB_REAL	Q(32 bit) = IN1(32 bit) – IN2(32 bit)	十进制数，带符号和小数，8 位（包括小数位）
SUB_UINT	Q(16 bit) = IN1(16 bit) – IN2(16 bit)	无符号十进制数，5 位

减法操作数

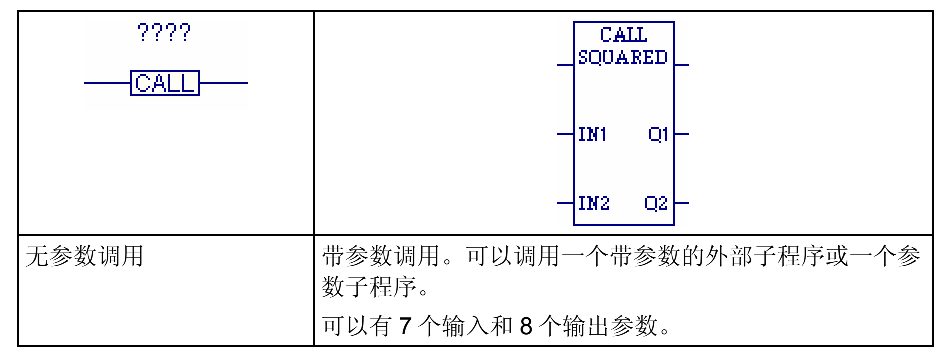
参量	描述	许用操作数	可选性
IN1	被减数，等式 IN1-IN2=Q 中减号左边的数	除 S, SA, SB, SC 外任何操作数	No
IN2	减数，等式 IN1-IN2=Q 中减号右边的数	除 S, SA, SB, SC 外任何操作数	No
Q	IN1-IN2 的结果。如果产生溢出，结果是带适当符号的最大值，能流断开。	除 S, SA, SB, SC 和常数外任何操作数	No

程序流程功能

程序流程功能限制程序执行或改变 CPU 执行应用程序的方式。

功能块	*助记符*	*描述*
子程序调用	CALL	调用子程序
注释	COMMENT	把一个文本解释放在程序中
结束主控继电器	ENDMCRN	嵌套结束主控继电器。表示在正常能量流情况下要执行的后续逻辑
逻辑结束	END	逻辑无条件结束。程序从第一梯级执行到最后梯级或 END 指令，无论先遇到哪个程序结束。
跳转	JUMPN	嵌套跳转。导致程序执行跳转到一个 LABELN 指出的指定存储单元。JUMPN/LABELN 对能相互嵌套。多个 JUMPN 能共有相同的 LABELN。
标号	LABELN	嵌套标号。指定一个 JUMPN 指令的目标位置。
主制继电器	MCRN	嵌套主控继电器。导致在 MCR 和其后的 ENDMCRN 之间所有的梯级在没有能流时执行。MCRN/ENDMCRN 对能互相嵌套。所有的 MCRN 能共有一个相同的 ENDMCRN。
连线	H_WIRE	为了完成能流传递，水平连接 LD 逻辑的一行元素。
	V_WIRE	为了完成能流传递，垂直连接 LD 逻辑的一列元素。

子程序调用



无参数调用	带参数调用。可以调用一个带参数的外部子程序或一个参数子程序。 可以有 7 个输入和 8 个输出参数。

当 CALL 功能块接收能流，它将使逻辑执行立即跳转到目的程序块，外部 C 子程序（带参数或无参数），或参数子程序并执行。该子程序执行结束后，控制立即返回在 CALL 指令之后的原调用点。

注意:

- 一个 CALL 功能块能在任何程序块中使用，包括_MAIN 块或一个带参数块。但不能在一个外部块中使用。
- 不能调用一个_MAIN 块。
- 执行调用之前，被调用的块必须存在。
- 一个已知块的调用和被调用的次数没有限制。
- 通过调用块本身可以形成递归子程序。当堆栈容量配置为默认值（64K），PLC 保证在“应用堆栈溢出”错误发生之前 8 个嵌套调用中最小一个调用。
- 当一个程序块、带参模块或外部 C 块的 Y0 参数返回 ON 时，CALL 向右传递能流，当返回为 OFF 时，CALL 不向右传递能流

注意： 每个块都有预设参数，Y0，在每次调用之前 CPU 把它设置为 1。Y0 能被块内逻辑控制并提供块的输出状态。

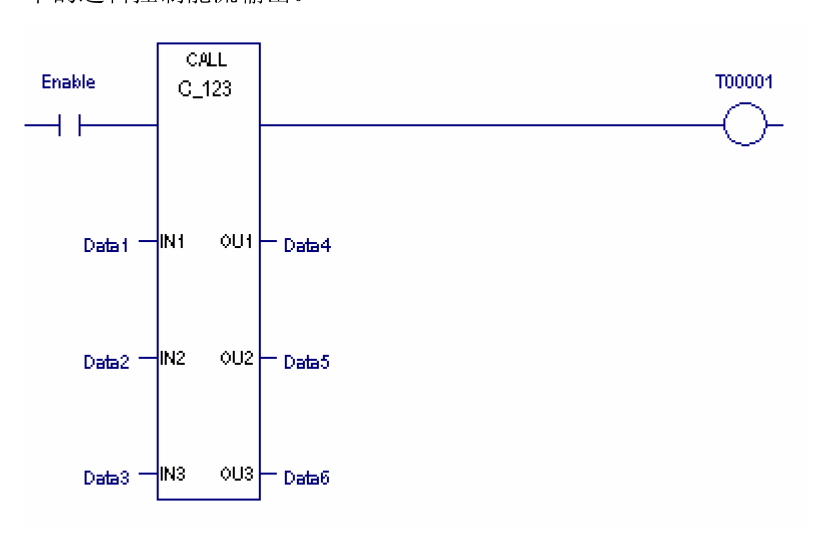
Call 操作数

参量	*描述*
块名称(????)	块名称，要跳转到的块名称。 不能调用_MAIN 块。 一个程序块或一个带参块能调用本身。
(只用于带参数块调用) 输入参数(0 – 7) 输出参数 (1 – 8)	**关于外部（C）块的注意：** ■ 必须定义每个外部 C 块参数的 TYPE, LENGTH,和 NAME。 ■ 有效数据类型，数据范围和每个参数的存储区域都在外部块的写文本中有规定。 ■ 对任何参数数据流都是允许的。 ■ 更多的信息见第 6 章的外部块部分。 **关于带参块的注意：** ■ 必须定义每个参数的 TYPE, LENGTH 和 NAME。CALL 指令中的有效操作数包括变量、流和间接参考。输入操作数也可以是常量。 ■ 如果一个形式参数是一个 BOOL 数组，且其长度是 16 的倍数，那么残留在字组存储器中的变量或数组可以作为一个操作数传递到带参数块中。例如，一个带参数块有一个数据类型为 BIT，长度为 48 的形式参数 Y1，可以给 Y1 送一个长度为 3 的字组。 ■ 对所有带参数块，BOOL 参数 Y0 是自动定义的，可以用于带参数块的逻辑中。当带参数块暂停且 Y0 是 ON，CALL 向右传递能流。如果 Y0 是 OFF，CALL 断开能流。 ■ 带参数块不需要输入和输出的数量相同。 ■ 附加说明，见第 6 章的“在带参数块中使用参数”

子程序调用举例

例 1

在下面的例子中，如果使能接通，名为 C_123 的 C 块执行。C_123 对基准地址 Data1, Data2 和 Data 3 的输入数据操作，产生数据放在基准地址 Data4, Data5 和 Data6。C_123 中的逻辑控制能流输出。



例 2

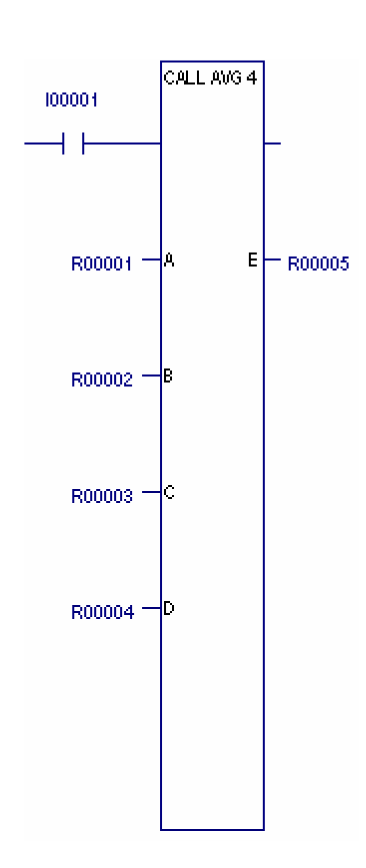
带参块对建立自定义功能块库是有用的。例如，如果有下面的等式：

E=(A+B+C+D)/4,

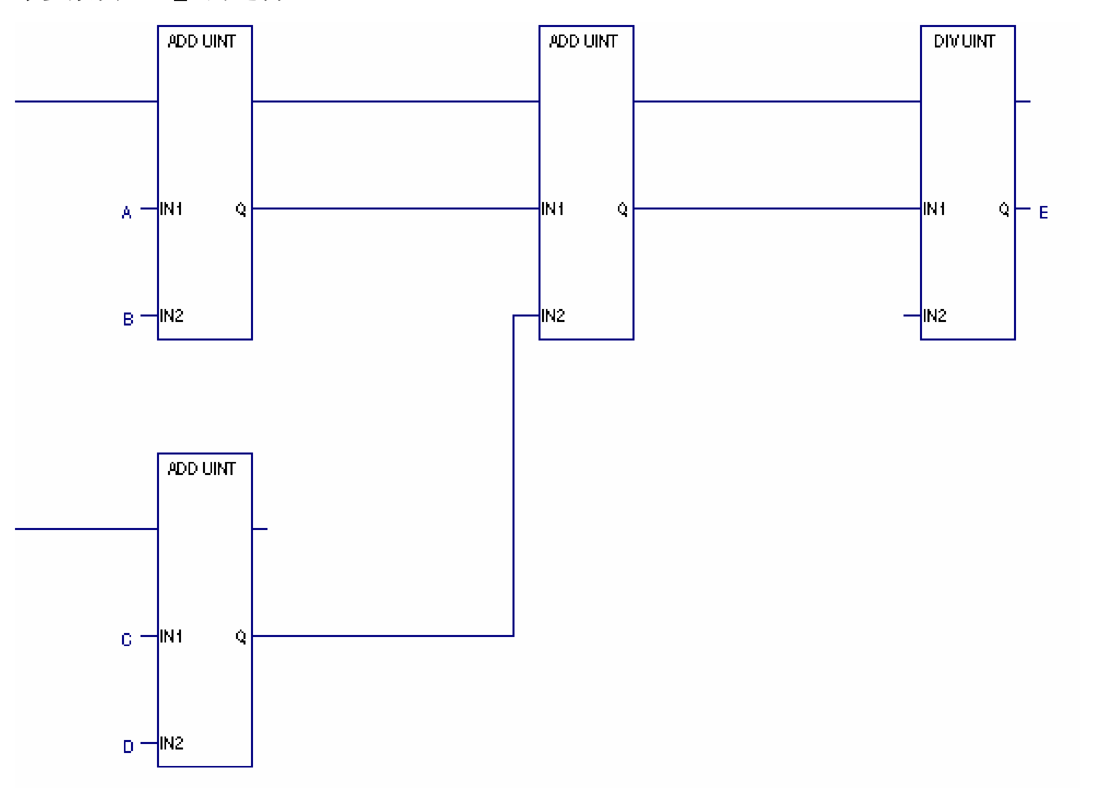
一个名为 AVG_4 的带参子程序能够如例中所示的被调用到右边。

在这个例子中，R00001, R00002, R00003, 和 R00004 中的数值的平均值将放在 R00005。

带参块的逻辑如下定义：



带参数块 AVG_4 的逻辑



注释

????

注释功能被用来在程序中加入一个文本解释。把一个注释指令插入 LD 逻辑中时，显示????。键入一个注释之后，头几个字被显示。

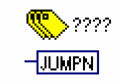
This is a comment. Comments have no effect on program execution.

在逻辑 Developer – PLC 软件中，可以设置注释模式选项为摘要或全文显示。

注意：

- 在 Microsoft ® Windows ® 9x 中，最大长度是 32,766 字符(0x7FFE)。在 Microsoft ® Windows 2000 ® 或 NT ®中，一个注释的最大长度是 2GB – 1 字节，总共 2,147,483,646 字符(0x7FFFFFFE)。
- 因为注释不能下载到 PLC 中，故可在线或离线编辑注释，效果相同。

跳转



助记符	*描述*	*总是相关*
JUMPN	跳转指令的嵌套形式	LABELN 指令

一个 JUMP 指令将旁路逻辑程序的一部分。程序在同一个块中指定的 LABELN 中连续执行。能量流直接从 JUMPN 跳转到由 LABELN 指定梯级。

当跳转激活时，在 jump 和 label 之间的任何功能块都不执行。在 JUMPN 和与其相关的 LABELN 之间的所有线圈都保持它们先前的状态。包括与定时器、计数器、锁存器和继电器相关联的线圈。

任何 JUMPN 能向前跳转也能向后跳转，也就是说，LABELN 既能在前面梯级中也能在后面梯级中。LABELN 必须在同一个块中。

注意：为了避免由向前或向后 JUMPN 指令建立一个死循环，一个向后 JUMPN 应该包含一条有条件的路径。

一个 JUMPN 指令的右端不连接任何指令或语句。

操作数

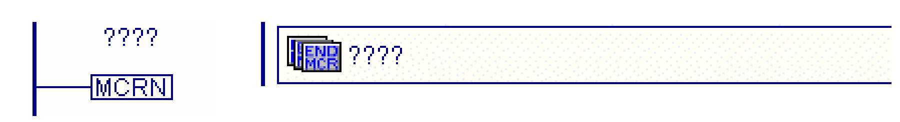
参量	*描述*	*可选性*
Label (????)	标号名；指派给目的 LABEL(N)的名字	No

操作说明

一个 JUMPN 与和它相关联的 LABELN 能放在程序的任何地方，只要 JUMPN / LABELN 在下面的范围内：

- 不与 MCRN / ENDMCRN 对的范围重叠
- 不与 FOR_LOOP / END_FOR 对范围重叠

主控继电器/结束主控继电器



助记符	描述	总是相关
MCRN	主控继电器嵌套形式	一个 ENDMCRN 指令
ENDMCRN	结束主控继电器嵌套形式	一个 MCRN 指令

MCRN

一个 MCRN 指令标志一个逻辑段的开始，执行该逻辑段时断开能流。一个 MCRN 段的结束必须由一个和 MCRN 同名的 ENDMCRN 作记号。在逻辑里 ENDMCRN 必须跟在相应的 MCRN 后面。

在激活的 MCRN 和相应的 ENDMCRN 之间的所有梯级执行时，有来自母线的负能流。和 MCRN 相关的 ENDMCRN 使得正常的程序执行重新开始，正的能流从母线进入。

关于一个主控继电器，在主控继电器范围内的功能块执行时，能流断开，线圈关断。

在一个激活的主控继电器范围内的子程序调用不执行。可是，在子程序内的任一定时器连续不断地计时。

一个 MCRN 后面的梯级可以是空的。

不象 JUMP 指令，MCRN 只能前移。在一个程序中，一个 MCRN 指令后面必须跟一个相应的 ENDMCRN 指令。

一个 MCRN 强加下列控制：

- 定时器不计时。TMR 类型重置。对于一个 ONDTR 功能块，累加器保持原来的值。
- 正常输出关断，取反输出接通。

注意: 当 MCRN 激活，扫描被它控制的逻辑，显示触点状态。如果不留心一个 MCRN 正控制逻辑，这看来是情形不妙。

MCRN 和相应的 ENDMCRN 可以放在程序中任何位置，和 MCRN / ENDMCRN 范围一样长。

- 在另一个 MCRN / ENDMCRN 范围内是完全嵌套，最大可到 256 级，或完全在另一个 MCRN / ENDMCRN 范围之外。
- 在 FOR_LOOP / END_FOR 范围内是完全嵌套，或完全在 FOR_LOOP / END_FOR. 范围之外。

EndMCRN

EndMCRN 指令标志着一个开始于一个 MCRN 指令的逻辑段的结束。当与 EndMCRN 相关的 MCRN 激活，EndMCRN 使程序执行返回，能流恢复。 。当与 EndMCRN 相关的 MCRN 不激活，。EndMCRN 指令没影响。

ENDMCRN 必须连接在母线上，在该梯级上 ENDMCRN 之前没有逻辑，ENDMCRN 无条件执行。

ENDMCRN 有一个名称，用来识别及联合相应的 MCRN 。ENDMCRN 功能块没有输出，在一个梯级中 ENDMCR 指令的后面没有任何指令或语句

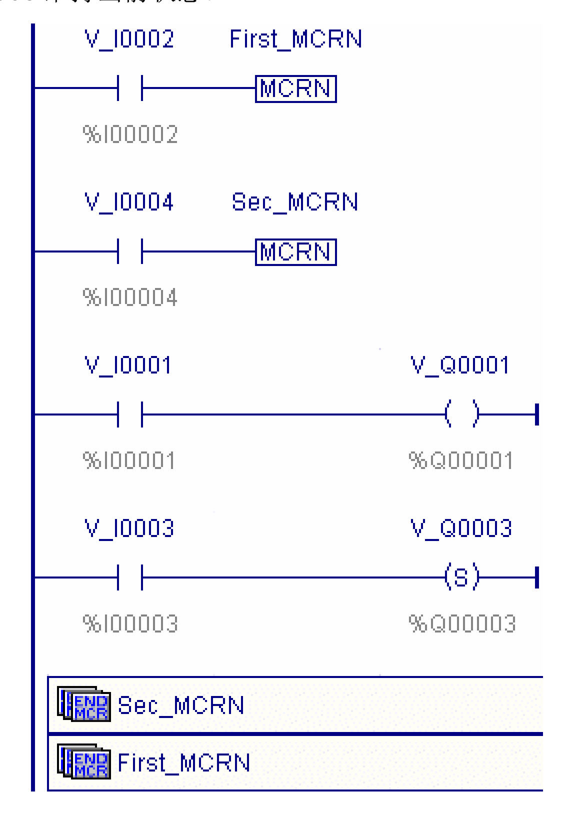
MCRN/ENDMCRN 操作数

MCRN 功能块有一个单独的操作数，识别 MCRN 的名称。这个名称和 ENDMCRN 指令一起再一次被用。MCRN 没有输出。

参量	*描述*	*可选性*
名称 (????)	和启动逻辑段的 MCRN 相关的名称	No

MCRN/ENDMCRN 举例

下面的例子展示了一个嵌套进名为“First_MCRN.”的 MCRN 的名为“Sec_MCRN” 的 MCRN。只要 V_I0002 触点允许能流进入 MCRN 功能块，程序执行，能流不进入线圈，一直执行到关联的 ENDMCRN。如果 V_I0001 和 V_I0003 触点为 ON，线圈 V_Q0001 关断，SET 线圈 V_Q0003 维持当前状态。



连线

H 水平和垂直线(H_WIRE 和 V_WIRE)被使用来连接功能块之间的 LD 逻辑的一行的元素。它们的目的是为了完成一个逻辑行从左到右的逻辑流。

一个垂直线传送在紧接它左边的元素的 BOOLEAN ON/OFF 状态到紧接它的右边元素。

一条垂直线可以在每一边与一条或更多条水平线相交。垂直线的状态是包括在它左边的垂直线的 ON 状态的 OR。垂直线的状态被复制到右边的附着的全部水平线上。

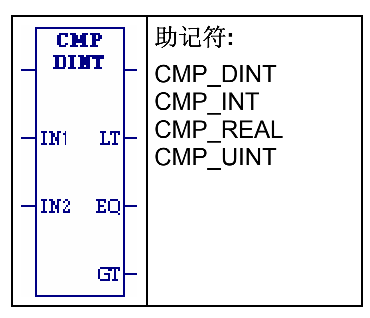
注意：连线能被用于数据流，但不能向左发送数据流。两个单独的数据流线路进入相同垂直线的左侧。

相关功能块

关系功能块比较相同数据类型的两个数值或决定一个数是否在给定的范围内。原值不受影响。

功能	*助记符*	*描述*
比较	CMP_DINT CMP_INT CMP_REAL CMP_UINT	比较两个数，IN1 和 IN2，助记符指定数据类型 ■ IN1 < IN2，LT 输出打开 ■ IN1 = IN2，EQ 输出打开 ■ IN1 > IN2，GT 输出打开
等于	EQ_DINT EQ_INT EQ_REAL EQ_UINT	检验两个数是否相等
大于或等于	GE_DINT GE_INT GE_REAL GE_UINT	检验一个数是否大于或等于另一个数
大于	GT_DINT GT_INT GT_REAL GT_UINT	检验一个数是否大于另一个数
小于或等于	LE_DINT LE_INT LE_REAL LE_UINT	检验一个数是否小于或等于另一个数
小于	LT_DINT LT_INT LT_REAL LT_UINT	检验一个数是否小于另一个数
不等于	NE_DINT NE_INT NE_REAL NE_UINT	检验两个数是否不等
范围	RANGE_DINT RANGE_DWORD RANGE_INT RANGE_UINT RANGE_WORD	检验一个数是否在另两个数给定的范围内

比较



助记符:

CMP_DINT
CMP_INT
CMP_REAL
CMP_UINT

当比较(CMP)功能块接收数据流，它将数值 IN1 跟 IN2 进行比较。

- 如果 IN1 < IN2, CMP 使 LT（小于）输出激活
- 如果 IN1 = IN2, CMP 使 EQ （等于）输出激活
- 如果 IN1 > IN2, CMP 使 GT （大于）输出激活

IN1 和 IN2 必须是相同的数据类型。CMP 比较下面类型的数据：DINT, INT, REAL 和 UINT。

提示：　比较两个不同数据类型的数值，首先使用转换功能块使数据类型相同。

当 CMP 接收能流，总是向右传递能流，除非 IN1 和/或 IN2 是非数。

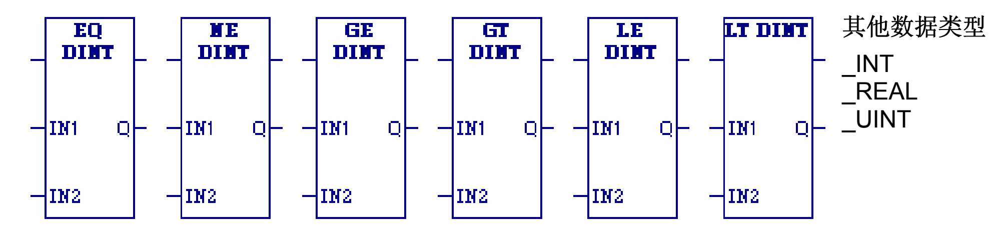
操作数

参数	描述	许用操作数	可选性
IN1	要比较的第一个数	除 S, SA, SB, SC 任何操作数	No
IN2	要比较的第二个数	除 S, SA, SB, SC 任何操作数	No
LT	I1 < I2 时输出 LT 激活	能流	No
EQ	I1 = I2 时输出 EQ 激活	能流	No
GT	I1 > I2 时输出 GT 激活	能流	No

范例

当%I00001 打开时，整数变量 SHIPS 与变量 BOATS 比较。内部线圈%M0001, %M0002,和 %M0003 设置为比较的结果。

等于，不等于，大于等于，大于，小于等于，小于



当关系功能块接收能量流，它比较输入 IN1 和输入 IN2。这些操作数必须是相同的数据类型。如果输入 IN1 和 IN2 相等，功能块向右传送能量，除非 IN1 与 IN2 同时为非数或有一个是非数。下面的关系功能块被使用来比较两个数。

功能块	*描述*	*关系说明*
EQ	等于	IN1=IN2
NE	不等于	IN1≠IN2
GE	大于等于	IN1≥IN2
GT	大于	IN1>IN2
LE	小于等于	IN1≤IN2
LT	小于	IN1<IN2

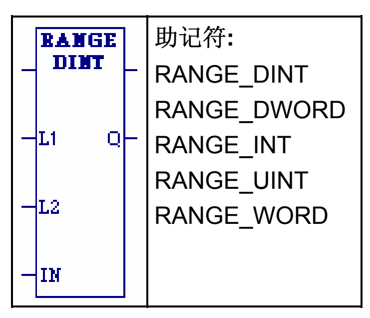
注意： 当这些关系功能块中的一个执行成功时，%S0020 位设为 ON。当任一个输入是非数，该位清零。是因为 NaN 在浮点数形式中有一个专用表示法，该表示法使其在任何功能块中都是可发觉的。在这些功能块的_DINT, _INT 或 _UINT 形式中没有这样的特性存在。如果在前面的 DINT, INT 或 UINT 操作中发生溢出，结果是有适当符号的最大可能值，能流断开。如果_DINT, _INT 或_UINT 操作反馈的是任何符号的最大可能值，它们不能确定是否是溢出值。前面操作的能流输出将需要被校验。

提示： 比较不同数据类型的数值，首先使用转换功能块使类型相同。关系功能块要求数据是下列类型之一：DINT, INT, REAL 或 UINT。

操作数

参量	*描述*	*许用操作数*	*可选性*
IN1	被比较的第一个数值；关系式左边的数值	除 S, SA, SB, SC 任何操作数	No
IN2	被比较的第二个数值；关系式右边的数值。IN2 必须与 IN1 数据类型相同	除 S, SA, SB, SC 任何操作数	No
Q	能流。如果关系式为真，Q 激活，除非 IN1 或 IN2 是非数。	能流	No

范围



当范围功能块激活，它将输入 IN 与操作数 L1 和 L2 限定的范围进行比较。L1 与 L2 中的任一个都可是上限或下限。当 L1 ≤ IN ≤ L2 或 L2 ≤ IN ≤ L1 时，输出参数 Q 设置为 ON(1).否则，Q 设为 OFF（0）。

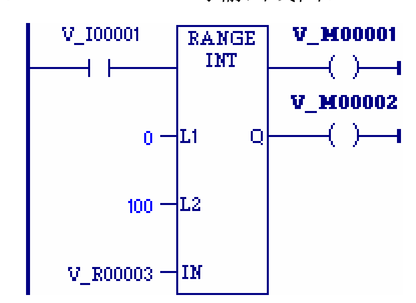
如果操作成功，它向右传送能流。

操作数

参量	描述	许用操作数	可选性
IN	与 L1 和 L2 限定的范围相比较的数值。必须与 L1 和 L2 数据类型相同	除 S, SA, SB, SC 任何操作数	No
L1	范围的开始点。可以是上限或下限。必须与 L1 和 L2 数据类型相同	除 S, SA, SB, SC 任何操作数	No
L2	范围的结束点。可以是下限或上限。必须与 L1 和 L2 数据类型相同	除 S, SA, SB, SC 任何操作数	No
Q	如果 L1 ≤ IN ≤ L2 或 L2 ≤ IN ≤ L1，Q 激活；否则，Q 关断。	能流	No

范例

当 RANGE_INT 从常开触点%I0001 接收能量流，它测定%R00003 中的数值是否包含在 0 到 100 的范围内。只有 0 ≤ %AI0050 ≤ 100 时输出线圈%M00002 打开。



计时器和计数器

这部分描述指令系统中的定时和计数功能块。

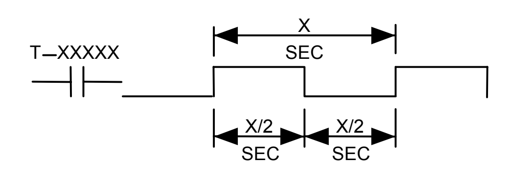
定时触点

PACSystems 有四个定时触点能用来提供到其他程序功能块能流的规则脉冲。定时触点以方形波形式每 0.01 秒、0.1 秒、1.0 秒和 1 分循环开和关。定时触点能够被一个外部通信设备读取以监控 CPU 的状态和通信线路。定时触点也经常被使用来点亮标志灯和发光二极管。

定时触点作为 T_10MS (0.01 s), T_100MS (0.1 s), T_SEC (1.0 s)和 T_MIN (1 min)的基准。定时触点表示%S 存储器中的指定存储单元。

#T_10MS	0.01s 定时触点	%S0003
#T_100MS	0.1 s 定时触点	%S0004
#T_SEC	1.0 s 定时触点	%S0005
#T_MIN	1.0 min 定时触点	%S0006

定时触点提供一个相当于开/关持续时间的脉冲。下面的时间图说明了这些触点的开/关持续时间。



警告

不要在要求精确测量总时间的应用软件中使用定时触点。这些类型的应用软件首选计时器，时基子程序和 PID 块。

CPU 基于一个自由运行的定时器更新定时触点基准，该计时器与 CPU 扫描开始没有关系。如果扫描时间与定时触点时钟保持同相，定时触点将一直显示相同的状态。例如，如果 CPU 是在一个扫描时间设定为 100ms 的固定扫描模式，T_10MS 和 T_100MS 位将不会触发。

注意： 对于定时触点在 PACSystems CPU、90-70 系列和 90-30 系列操作中的不同的简述见附录 C 中的“LD 功能块的不同”。

定时器和计数器功能块

三种类型的定时器功能块包含一个开延时定时器，一个关延时定时器和一个启动－复位定时器。与这些功能块相关联的数据通过动力循环保持。

功能块	助记符	描述
延时关定时器	OFDT_HUNDS OFDT_SEC OFDT_TENTHS OFDT_THOUS	当能流输入打开时定时器的当前值（CV）重设为 0。当能流关时 CV 增加。当 CV=PV（预置值），能流不再向右传送直到能流输入再次打开。
On Delay Stopwatch Timer	ONDTR_HUNDS ONDTR_SEC ONDTR_TENTHS ONDTR_THOUS	保持型延时定时器。当它接收能流时计时，在能流停止时保持它的值
延时开定时器	TMR_HUNDS TMR_SEC TMR_TENTHS TMR_THOUS	一般延时定时器。当它接收能量时计时，能流停止时重设为 0。
减计数器	DNCTR	从预置值倒计数。一旦 CV≤ 0 输出接通。
增计数器	UPCTR	计数直到一个指定值。一旦 CV≥PV 输出接通。

注意：在带参模块中编制定时器时程序特别注意。详见8-**错误！未定义书签。** 的“在带参模块中使用定时器”。

在定时器和计数器功能块中要求的数据

每个定时器和计数器使用 %R, %W, %P, %L 的一元的三字数组或符号存储器来存储下面的信息：

当前值 (CV)	Word 1
预设值 (PV)	Word 2
控制字	Word 3

当加入一个定时器时，必须加入一个三字数组的开始地址（寄存器三个字的块）。

警告

不要使用两个连续的字（寄存器）作为两个定时器或计数器的开始地址。如果寄存器地址重合，逻辑 Developer - PLC 不会检查或发出警告。如果将第二个定时器的当前值代替前一个定时器的预置值，定时器将不会工作。

Word1：当前值(CV)

警告

第一个字（**CV**）能够读取但不应该写入，否则功能块可能不能正常工作。

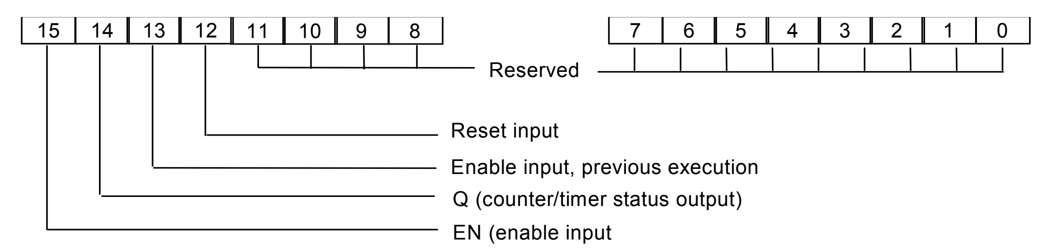
Word 2:预置值(PV)

当预置值操作数是一个变量，它一般设置为一个与在定时器或计数器的三字数组中第二个字不同的地址。

- 如果使用一个不同的地址，直接改变第二个字，改变将会没有作用，因为 PV 将会改写第二个字。
- 如果 PV 操作数和第二个字使用相同的地址，当定时器或计数器运行时可以改变第二个字中 PV 的值，并且改变是有效的。

Word 3:控制字

控制字存储布尔型与定时器或计数器相关的输入/输出状态，如下面的图表所示：



警告

第三个字（控制字）能够读取但不应该写入；否则，功能块将不能工作。

注意：

- 计数器中不使用位 0 到 13
- 定时器精度使用位 0 到 13

在带参模块中使用 OFDT, ONDTR 和 TMR

在 PACSystems 带参模块中编制定时器程序要特别注意。按下面的方针和规则来编制在带参模块中定时器程序来实现实时跟踪。如果不遵循此处描述的方针和规则，在带参模块中定时器功能块的操作是未定义的。

注意： 程序软件不会强迫遵循这些规则。使用者确保遵循这些规则。

定时器功能块的最佳用处是每次扫描用一个精确基准地址准确调用它。对带参模块，重要的是使用适当的带定时器功能的基准存储器和调用带参模块适当次数。

找到源块

源块是_MAIN 块或类型块的最底下的逻辑块，这个逻辑块在调用树里出现的在带参模块中之上。为了确定一个已知带参块的源块，要确定哪个块调用该带参模块。如果调用块是_MAIN 或类型块，，它就是源块。如果调用块是任何其他类型（带参模块或功能块），对调用这个块的块应用相同的测试。继续回退调用树直到_MAIN 块或一个类型块被找到。这是带参块的源块。

在带参块中编制 OFDT, ONDTR 和 TMR 程序

根据在程序逻辑中是否不止一个地方使用带参数块来决定应用不同的方针和规则

从一个块中调用的带参块

如果包含一个定时器的带参数块将只从一个逻辑块中被调用，遵循这些规则：

1. 每执行一次源块，调用一次带数参块。
2. 为定时器选择一个基准地址，这个定时器不能在任何其他地方使用。基准地址可以是%R, %P, %L, %W 或符号地址。

注意： %L 存储器和类型块的源块可用的%L 存储器变量是相同的。当源块是_MAIN 时%L 存储器相当于%P 存储器。

从多个块中调用的带参块

当从多个块中调用带参块时，分开对带参块的每次调用使用的定时器基准地址是必要的。遵循下面这些规则和方针：

1. 带参数块所在的源块每执行一次。调用带参模块一次
2. 为定时器基准存储器选择一个%L 基准或带参块形式参数。不要使用%R, %P, %L, %W 或符号存储器基准。

注意：

- 强烈推荐选择一个%L 存储单元，该存储单元从带参模块的源块继承得来。除_MAIN 块外每个源块都有它自己的%L 存储器空间，_MAIN 模块有一个%P 存储器区域。当_MAIN 块调用另一个块时，由_MAIN 块的映射%P 都被调用模块当作%L 映射访问。

如果使用一个带参块形式参数（字数组），对应这个形式参数的实在参数必须是一个%L, %R, %P, %W 或符号 基准地址。如果实在参数是一个%R, %P, %W 或符号 基准地址，每一个源块必须使用唯一的基准地址。

递归

如果使用递归（就是说，如果有一个直接或间接调用它本身的块）并且带参块包括 OFDT, ONDTR 或 TMR，必须遵循额外的两条规则：

- 编制源块次序使其能在对本身递归调用之前调用带参块；
- 不要编制直接调用带参块本身的程序。

在功能子程序中使用定时器

功能子程序是带参数和实例数据的自定义逻辑子程序。这些子程序和其他子程序的详细说明，见第 6 章。

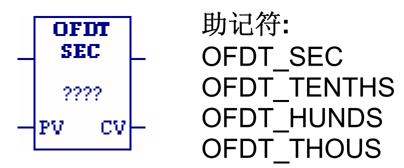
当一个定时器功能块在一个功能子程序中出现，并且一个部分变量被使用来控制一个定时器子程序时，定时器的动作可能与预期不匹配。如果功能块的多个实例在一个逻辑扫描期间被调用，只有第一次执行实例会更新它的定时器。如果一个不同的实例接着被执行，它的定时器值将保持不变。

在一次逻辑扫描期间对一个功能块的多次调用的情况下，只有第一次调用会向它的定时器功能块增加总时间。这个动作和在一个正常程序块的定时器的动作是匹配的。

范例：

一个功能块被定义为对一个定时器功能块使用一个成员变量。该功能块的两个实例 timer_A 和 timer_B 被建立。在每次逻辑扫描期间，timer_A 和 timer_B 都被执行。然而，只有在 timer_A 中的成员变量被更新，timer_B 中的成员变量一直保持为 0。

延时关定时器



助记符:
OFDT_SEC
OFDT_TENTHS
OFDT_HUNDS
OFDT_THOUS

当能流关断时，关延时定时器(OFDT)开始计时；而当能流开时，定时器的当前值重设为0。当未达到指定的时间间隔 PV（预置值）时，OFDT 传送能量。

有下列计时单位：

- 秒
- 0.1 秒
- 0.01 秒
- 0.001 秒

PV 的范围是 0 到+32,767 时间单位。如果 PV 超出范围，不影响定时器字 2。这个定时器保持在失电时状态；上电时不自动初始化。

当 OFDT 接收能流时，CV 设置为 0，定时器向右传送能量。OFDT 接收能流时输出保持。

每次 OFDT 被调用，它的能流输入关断，CV 被更新，以反映定时器复位后的总时间。OFDT 持续向右传送能量到直到 CV 等于或超过 PV。当 CV 等于或超过 PV 时，OFDT 停止向右传递能流到并停止计时。如果 PV 是 0 或负数，第一次能流输入关闭时被调用，定时器停止向右传递能流。

当功能块再次接收能流时，CV 重设为 0。

注意：

- 使用一个 OFDT 功能块的最好方式就是，每次扫描从一个特别的基准地址调用一次。每次扫描不要从一个特别的基准地址调用 OFDT 多于一次（否则会导致不适当的计时）。

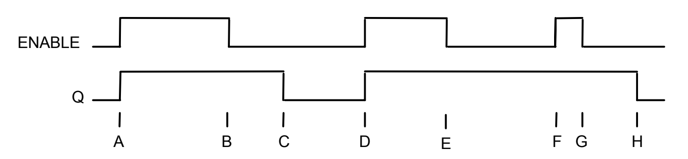
  当一个 OFDT 出现在一个程序块中，每次扫描就累计一次时间。相同扫描期内后来调用该程序块将对 OFDT 没有影响。

- 不要在两个不同子程序中编入带相同的基准地址的 OFDT 功能块。不应该编一个围绕定时器功能的跳转的程序。如果你使用递归（也就是说子程序直接或间接调用本身），要编制程序块以便在定时器递归调用本身之前调用定时器。
- 在带参数块内部使用定时器的相关信息，见8-**错误！未定义书签。**。

如果前次扫描时间大于 PV,OFDT 终止（关断能流）不接收能流的第一次扫描。

- 在一个每次扫描都不调用的程序块中使用 OFDT，除非定时器复位，不然在程序块内的定时器计时。这意味着 OFDT 功能块象一个比主程序块中定时器扫描更慢的定时器一样运行。对那些长时间的非活动的程序块，对 OFDT 编程时应该考虑到这种追赶特性。例如，如果一个程序块中的一个定时器被复位，该程序块四分钟不被调用（非活动的），当程序块被调用时，四分钟时间已经累积进去。如果使能输入关闭，这四分钟被应用到定时器中（也就是说，CV 设为 4 分钟）。

时序图



A. ENABLE 和 Q 都变高电平，定时器复位(CV = 0)

B. ENABLE 变低电平，定时器开始计时

C. CV 达到 PV，Q 变低电平，定时器停止计时

D. ENABLE 变高电平，定时器复位(CV = 0)

E. ENABLE 变低电平，定时器开始计时。

F. ENABLE 变高电平，在 CV 有机会达到 PV 之前定时器复位(CV = 0)（图标没有缩放）

G. ENABLE 变低电平，定时器开始计时

H. CV 达到 PV，Q 变低电平，定时器停止计时

操作数

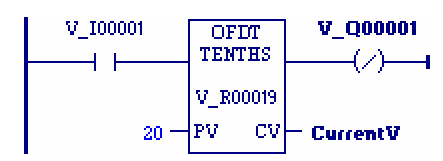
警告

不要用其他指令使用 Address, Address+1,或 Address+2 地址。基准地址重叠将导致不确定的定时器操作。

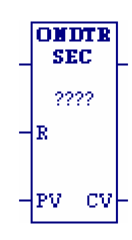
参量	*描述*	*许用操作数*	*可选性*
Address (????)	一个三字字数组的开始地址 Word 1: (CV)当前值 Word 2: (PV) 预置值 Word 3:控制字	R, W, P, L 符号地址	No
PV	预设值，当定时器激活或复位时使用。0 ≤ PV ≤ +32,767 如果 PV 超出范围，对字 2 无影响。	除 S, SA, SB, SC 外任何操作数	Optional
CV	定时器的当前值	除 S, SA, SB, SC,和常量外任何操作数	Optional

OFDT OFDT 举例

输出动作是由取反输出线圈来改变的。在这个电路中，只要触点%I0001 是闭合的，OFDT 定时器关取反输出线圈%Q0001。%I0001 打开之后，%Q0001 延时 2 秒之后打开。



跑表型延时开定时器



助记符:
ONDTR_SEC
ONDTR_TENTHS
ONDTR_HUNDS
ONDTR_THOUS

跑表型延时开定时器(ONDTR)接收能流时，ONDTR 计时；而当能流停止时，它的值保持不变。有以下的计时单位：

- 秒
- 0.1 秒
- 0.01 秒
- 0.001 秒

量程是 0 到+32,767 时间单位。这个定时器的状态在失电时保持不变，上电时不自动初始化。

当 ONDTR 第一次接收能流，ONDTR 开始计时（当前值 CV）。当 CV 等于或超过预置值(PV)，输出 Q 激活，不管能流输入的状态。

在定时器连续接收能流时，定时器连续计时直到 CV 等于最大值(+32,767 时间单位)。一旦达到了最大数值，CV 值保留，Q 保持激活状态，不管使能输入的状态如何。

当送到定时器的能流停止时，CV 停止计时并保持。如果输出 Q 已经激活则保持激活状态。当 ONDTR 再次接收能流，CV 从上次保留值开始再次计时。

当复位（R）接收能流，PV 不等于 0，CV 重设为 0，输出 Q 不激活。

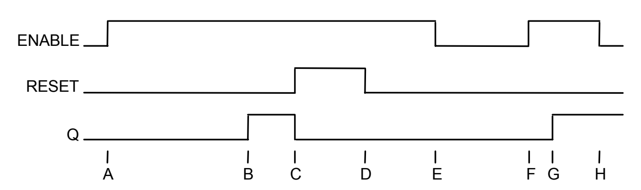
注意： 如果 PV 等于 0，定时器打开，定时器输出激活。随后把定时器的使能关闭或复位定时器，对定时器的输出没有影响，定时器保持激活状态。

当 CV 大于等于 PV 时，ONDTR 向右传递能流。由于上电时输出能流状态没有自动初始化，能流状态保持失电时的状态。

注意：

- 使用一个 ONDTR 功能块的最好方式就是，每次扫描从一个特别的基准地址调用一次。每次扫描不要从一个特别的基准地址调用 ONDTR 多于一次（否则会导致不适当的计时结果）。当一个 ONDTR 出现在一个程序块中，每次扫描就累计一次时间。相同扫描期内后来调用该程序块将对 ONDTR 没有影响。不要在两个不同子程序中编入带相同的基准地址的 ONDTR 功能块。不应该编一个围绕定时器功能的跳转的程序。如果你使用递归（也就是说子程序直接或间接调用本身），要编制程序块以便在定时器递归调用本身之前调用定时器。
- 在带参数块内部使用定时器的相关信息，见8-错误！未定义书签。。
- 如果前次扫描时间大于 PV, ONDTR 终止（能流向右传递）已经激活的，没有复位的第一次扫描。
- 在一个每次扫描都不调用的程序块中使用 ONDTR，除非定时器复位，不然在程序块内的定时器计时。这意味着 ONDTR 功能块象一个比主程序块中定时器扫描更慢的定时器一样运行。对那些长时间的非活动的程序块，对 ONDTR 编程时应该考虑到这种追赶特性。例如，如果一个程序块中的一个定时器被复位，该程序块四分钟不被调用（非活动的），当程序块被调用时，四分钟时间已经累积进去。如果使能输入接通且复位输入是 OFF，这四分钟被应用到定时器中（也就是说，CV 设为 4 分钟）。

时序图



A. ENABLE 变高电平，定时器开始计时。
B. 当前值（CV）达到预置值（PV）时，Q 变高电平。定时器连续累积时间直到 ENABLE 变低电平，RESET 变高电平或当前值变成等于最大值。
C. RESET 变高电平，Q 变低电平，累积时间复位（CV=0）。
D. RESET 变低电平，定时器接着重新开始累积时间，如 ENABLE 是高电平一样。
E. ENABLE 变低电平，定时器停止累积时间。累积的时间保存不变.
F. ENABLE 重新变高电平，定时器继续累积时间。
G. CV 变成等于 PV，Q 变高电平。定时器继续累加时间直到 ENABLE 变低电平，RESET 变高电平或 CV 变成等于最大值。
H. ENABLE 变低电平，定时器停止累加时间。

跑表型延时开定时器的操作数

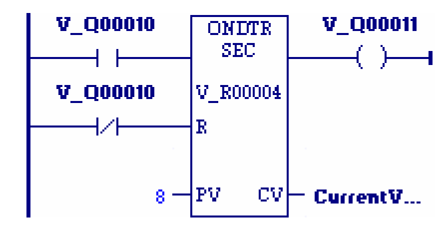
警告

不要用其他指令使用 **Address, Address+1,**或 **Address+2** 地址。基准地址重叠将导致不确定的定时器操作。

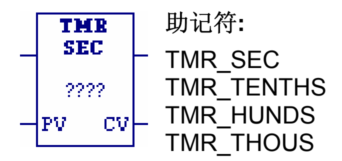
参量	*描述*	*许用操作数*	*可选性*
Address (????)	一个三字字数组的开始地址 Word 1: (CV)当前值 Word 2: (PV) 预置值 Word 3:控制字	R, W, P, L，符号地址	No
R	R 开时，重设当前值（字 1）为 0	能流	Optional
PV	预设值，当定时器激活或复位时使用。0 ≤ PV ≤ +32,767 如果 PV 超出范围，对字 2 无影响	除 S, SA, SB, SC 外任何操作数	Optional
CV	定时器的当前值	除 S, SA, SB, SC 和常量外任何操作数	Optional

跑表型延时开定时器举例

一个跑表型延时开定时器用来发生一个在%Q0010 打开 8 秒后打开的信号（%Q0011），当%Q0010 关时，信号关。



延时开定时器



助记符:

TMR_SEC
TMR_TENTHS
TMR_HUNDS
TMR_THOUS

当延时开定时器(TMR)接收能流时，定时器计时；而当能流停止时，TMR 回 0。只要定时器接收能，在指定间隔时间 PV（预置值）达到之后，定时器传递能流。

PV 的范围是 0 到+32,767 个时间单位。如果 PV 超出了范围，其对定时器字 2 无影响。定时器的状态在失电时保持，上电时不自动初始化。

有以下计时单位：

- 秒
- 0.1 秒
- 0.01 秒
- 0.001 秒

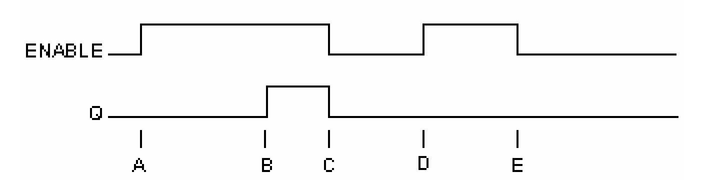
当 TMR 在能流输入关断时被调用，它的当前值（CV）重设为 0，定时器不向右传递能流。每次能流输入打开时 TMR 被调用，CV 更新为自定时器复位以来累计时间。当 CV 达到 PV 时，定时器功能块向右传送能流。

注意:

- 使用一个 TMR 功能块的最好方式就是，每次扫描从一个特别的基准地址调用一次。每次扫描不要从一个特别的基准地址调用 TMR 多于一次（否则会导致不适当的计时结果）。当一个 TMR 出现在一个程序块中，每次扫描就累计一次时间。相同扫描期内后来调用该程序块将对 TMR 没有影响。不要在两个不同子程序中编入带相同的基准地址的 TMR 功能块。不应该编一个围绕定时器功能的跳转的程序。如果你使用递归（也就是说子程序直接或间接调用本身），要编制程序块以便在定时器递归调用本身之前调用定时器。
- 在带参数块内部使用定时器的相关信息，见**8-错误！未定义书签。**。
- 如果前次扫描时间大于 PV, TMR 终止（能流向右传递）已经激活的、没有复位的第一次扫描。。

在一个每次扫描都不调用的程序块中使用 TMR，除非定时器复位，不然在程序块内的定时器计时。这意味着 TMR 功能块象一个比主程序块中定时器扫描更慢的定时器一样运行。对那些长时间的非活动的程序块，对 TMR 编程时应该考虑到这种追赶特性。例如，如果一个程序块中的一个定时器被复位，该程序块四分钟不被调用（非活动的），当程序块被调用时，四分钟时间已经累积进去。如果使能输入接通且复位输入是 ON，这四分钟被应用到定时器中（也就是说，CV 设为 4 分钟）

时序图



A. ENABLE 变高电平，定时器开始累积时间

B. CV 达到 PV；Q 变高电平，定时器继续累积时间

C. ENABLE 变低电平，Q 变低电平；定时器停止累积时间，CV 清零

D. ENABLE 变高电平；定时器开始累积时间

E. ENABLE 在当前值到达 PV 之前变低电平；Q 保持低电平，定时器停止累积时间并清零。

延时定时器开的操作数

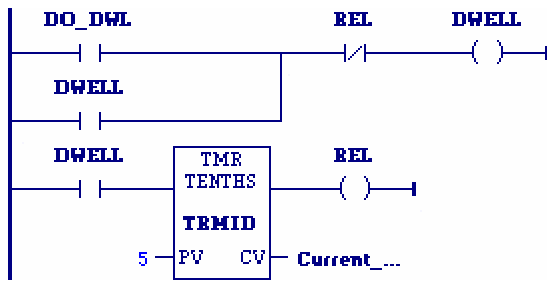
警告

不要用其他指令使用 Address, Address+1,或 Address+2 地址。基准地址重叠将导致不确定的定时器操作。

参量	*描述*	*许用操作数*	*可选性*
Address (????)	一个三字字数组的开始地址 Word 1: (CV)当前值 Word 2: (PV) 预置值 Word 3:控制字	R, W, P, L 符号地址	No
PV	预设值，当定时器激活或复位时使用。0 ≤ PV ≤ +32,767 如果 PV 超出范围，对字 2 无影响。	除 S, SA, SB, SC 外任何操作数	Yes
CV	定时器的当前值	除 S, SA, SB, SC,和常量外任何操作数	Yes

延时开定时器举例

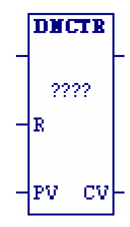
一个带地址的延时开定时器 TMRID 被使用来控制线圈打开的时间的长度。这个线圈已经被分配到变量 DWELL 中。当正常打开（瞬时）触点 DO_DWL 打开，线圈 DWELL 激活。



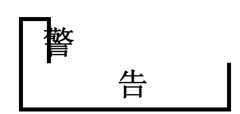
线圈 DWELL 触点保持线圈 DWELL 激活（当触点 DO_DWL 被释放），并打开定时器 TMRID。当 TMRID 达到它的 0.5s 的预置数值，线圈 REL 激活，中断线圈 DWELL 的自锁条件。触点 DWELL 中断到 TMRID 的能流，重设它的当前值并使线圈 REL 失电。回路准备好为另一次的触点 DO_DWL 瞬时激活。

减计数器

减计数器(DNCTR)功能模块从预置值递减计数。最小的预置值(PV)为 0，最大的预置值为+32767。当当前值(CV)到达最小值-32768，它将保持在那里不变直到复位。当 DNCTR 复位，CV 被置为 PV。当能量流输入从 OFF 变为 ON，CV 开始以 1 为单位递减。当 CV ≤ 0 时，输出为 ON



当失电时，DNCTR 的输出状态 Q 被保持；在得电时不会发生自动初始化。



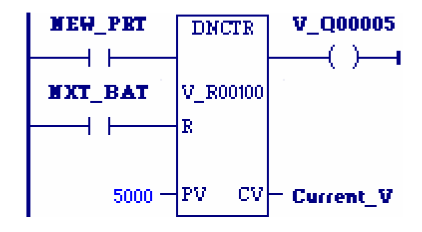
不要和其他指令一起用减计数器的地址。重叠的地址将引起不确定得计数器操作。

操作数

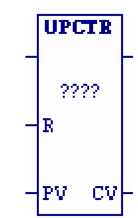
参数	*描述*	*许用操作数*	*可选性*
Address (????)	三个字字组的开始地址 Word 1: 当前值 (CV) Word 2: 预置值 (PV) Word 3: 控制字	R,W,P,L,符号地址	No
R	当 R 接收到能量流，它将重置 CV 为 PV	能流	No
PV	当计数器激活或者复位，复制进 word 2 的预置值。0 ≤PV ≤ 32,767。如果 PV 超出范围，word 2 不能重置。	除了 S,SA,SB,SC 外任何操作数的所有	No
CV	计数器的当前值。	除了 S,SA,SB,SC 和常数外任何操作数的所有	No

减计数器举例

在%Q00005 被激活前，DNCTR 从 5000 开始递减计数。



增计数器



增计数器功能模块(UPCTR)从预置值(PV)递增计数。计数的范围为 0 到 32767。当当前值(CV)到达 32767，值将保持直到复位。当 UPCTR 重置为 ON,CV 重置为 0。每次当能量流从 OFF 转换为 ON，CV 增加 1。CV 能增加到超过 PV。只要 CV ≥ PV，则输出为 ON。输出 Q 保持 ON 直到 R 输入接收到能量流来重置 CV 为 0。

在失电时 UPCTR 的状态保持，得电时不会发生自动初始化。

操作数

警告

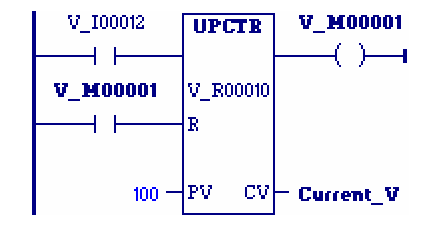
不要和其他指令一起用增计数器的地址。重叠的地址将引起不确定得计数器操作。

操作数

参数	*描述*	*许用操作数*	*可选性*
Address (????)	三个字字组的开始地址 Word 1: 当前值 (CV) Word 2: 预置值 (PV) Word 3: 控制字	R,W,P,L,符号地址	No
R	当 R 为 ON 时，计数器的 CV 置 0。	能流	No
PV	当计数器激活或者复位，复制进 word 2 的预置值。0 ≤PV ≤ 32,767。如果 PV 超出范围，不影响 word 2。	除了 S,SA,SB,SC 外任何操作数的所有	No
CV	计数器的当前值。	除了 S,SA,SB,SC 和常数外任何操作数的所有	No

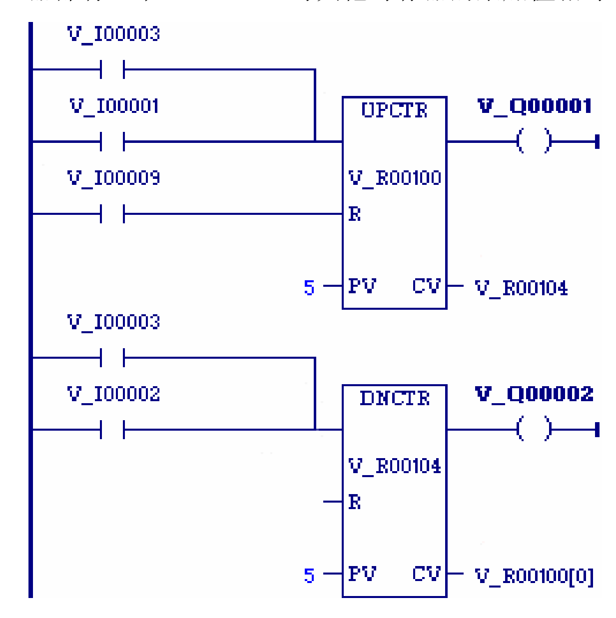
增计数器举例

每次当%I0012 从 OFF 转换为 ON 时，增计数器增加 1；只要 CV 超过 100 则线圈%M0001 被激活。只要%M0001 为 ON，计数器置为 0.



增计数器和减计数器举例

这个例子用了增/减计数器对，积累值和当前值共用一个寄存器用。当零件进入存储区，增计数器增加 1，零件的当前值增加 1。当一个零件离开存储区，减计数器减少 1，存货区的值减少 1。为了避免共用寄存器冲突，两个计数器用了不同的寄存器地址，但是每一个计数器都有一个 CV 地址，与其他寄存器的累加值相等。.

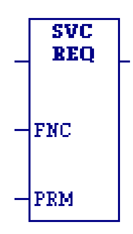


第九章 服务请求功能

本章介绍如何使用服务请求(SVC_REQ) 功能,用 SVC_REQ 去请求下面一个特别的控制系统服务:

*CPU 中冗余申请的服务请求信息,参考 *PAC 系统热备份 CPU 冗余用户指南*, GFK-2308

SVC_REQ 功能操作数



当 SVC_REQ 接收能流, 它请求 CPU 执行由 FNC 操作数确定的特别服务. SVC_REQ 的参数设置在参数块中, 它们以 PRM 操作数确定的基准地址为起始值,，必需的 16 位基准地址数目取决于被请求的特别 PLC 服务类型.参数块用于储存功能块的输入和输出量.。SVC_REQ 传递能流，除非指定了错误的功能块数,错误的参数或超出范围基准地址。 不同的特殊 SVC_REQ 功能有另外的引起失败的原因

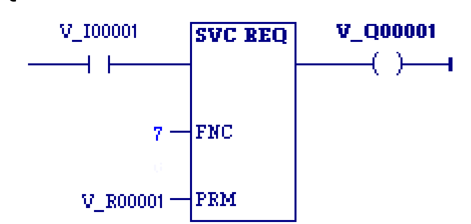
操作数

注意: 间接引用可用于所有的寄存器基准(%R, %P, %L, %W, %AI, 和 %AQ).

操作数	*数据类型*	*记忆区*	*描述*
FNC	INT 变量或常量	除了%S - %SC 以外所有区域	功能数，服务请求数。常数或确定请求服务的基准
PRM	WORD 变量	除了%S - %SC 以外所有区域	请求服务的参数块中的第一个字，连续的 16 位存储单元 存储另外的参数

例

当使能输入%I0001 为 ON, SVC_REQ 的功能数 7 和始于%R0001 的参数块被调用,. 如果操作成功，那么输出线圈%Q0001 置为 ON.



SVC_REQ 1: 改变/读固定扫描定时器

使用 SVC_REQ 的功能 1:

- 禁止固定扫描模式
- 固定扫描模式使能和使用旧的固定扫描定时器值
- 固定扫描模式使能和使用新的固定扫描定时器值
- 仅设置新的固定扫描定时器值
- 读固定扫描模式状态和定时器值.

参数块有两个字的长度用于输入和输出

SVC_REQ 在以下情况下不能成功执行:：

- 作为被请求的操作，输入的数字不是 0，1，2 或 3，。
- 扫描时间值大于 2550 ms (2.55 秒)
- 固定扫描时间激活，但定时器值没有程序化或是一个旧值 0。

使固定扫描模式不激活:：

带这个参数块上进入 SVC_REQ 1 :

地址	0
地址+ 1	忽视

固定扫描模式激活和使用旧的定时器值

使用此参数块进入 SVC_REQ 1 :

地址	1
地址+ 1	0

如果定时器的值不存在，输入 0 导致功能块置输出 OK 为 OFF。

固定扫描模式激活和使用新的定时器值

使用此参数块进入 SVC_REQ 1 :

地址	1
地址 + 1	新的定时器值 **注意**：如果定时器的值不存在，输入 0 导致功能块置输出 OK 为 OFF

改变定时器值而不改变选择的扫描模式状态

带这个参数块上进入 SVC_REQ 1:

地址	2
地址 + 1	新的定时器值

只读而不改变定时器的当前状态和值

使用此参数块进入 SVC_REQ 1 :

地址	3
地址+ 1	忽略

输出

SVC_REQ 1 在相同的参数块基准地址送回定时器状态和值

地址	0 = 正常扫描 1 = 固定扫描
地址 + 1	当前定时器值

如果字地址 + 1 包含了十六进制值 FFFF，那么定时器值没有被程序化 。

例

如果触点 OV_SWP 置为 1， 固定扫描定时器被读，定时器以两毫秒为单位增加，新的定时器值送回 CPU，参数块在存储单元%R3050 中。 该例中，逻辑使用的是离散的内部线圈%M0001 作为一个中间单元，保持梯形图第一行的成功执行的结果。任何扫描在 OV_SWP 没有置位时，%M00001 都是关断的。

OV_SWP
MOVE WORD
3
1 IN Q V_R03050
SVC REQ
1 FNC
V_R03050 PRM
ADD INT
V_R03051 IN1 Q V_R03051
2 IN2
V_M00001
V_M00001
MOVE WORD
1
1 IN Q V_R03050
SVC REQ
1 FNC
V_R03050 PRM

SVC_REQ 2: 读窗口模式和时间值

用 SVC_REQ 2 得到当前窗口模式和控制器通讯窗口、底板通讯和后台任务窗口的时间值

输出

地址	*窗口*	*高字节*	*低字节*
地址	控制器通讯窗口	模式	值（ms ）
地址+1	底板通讯窗口	模式	值（ms ）
地址+2	后台窗口	模式	值（ms ）

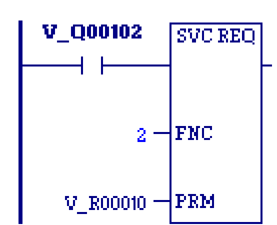
注意: 当时间值为零时，窗口被禁止。

模式值

模式名	*值*	*描述*
限制模式	0	窗口的执行时间被限于各自的缺省值，或使用 SVC_REQ 3 为控制器通讯窗口定义的值，或使用 SVC_REQ 4 为系统通讯窗口定义的值。当窗口没有更多的任务执行时将终止。
固定模式	1	每个窗口将运行在一个“运行到完成”模式， CPU 将在三个窗口之间交替，所用的时间等于各个窗口时间值之和。如果有一个窗口是固定模式，其他两个窗口将自动地置为固定模式。如果 CPU 运行在固定窗口模式，特别的窗口执行时间不是由相关的 SVC_REQ 功能来定义的.。窗口缺省时间用于固定窗口的时间计算。
运行到完成模式	2	不管和特殊窗口相关的窗口时间, 也不管缺省或由一个服务请求定义的时间，窗口将运行直到窗口内所有的任务完成.

例子

当%Q00102 被置位, CPU 把窗口的当前时间值放在开始于存储单元%R0010 的参数块中。



SVC_REQ 3: 改变控制器通讯窗口模式

用 SVC_REQ 3 去改变控制器通讯窗口模式和定时器值。当功能被调用，改变将在 CPU 的下一扫描期间发生

参数块有一个字长。

SVC_REQ 3 向右传递能流，除非选择的模式不是 0（限制）和 2（运行到完成）。

使控制器通讯窗口不激活:

使用此参数块进入 SVC_REQ 3:

地址	*高字节*	*低字节*
地址	0	0

重激活到或改变控制器通讯窗口模式:

使用此参数块进入 SVC_REQ 3:

地址	*高字节*	*低字节*
地址	模式: 0 = 限制 2 =运行到完成	1ms ≤ 值 ≤ 255ms， 增量 1ms

例子

当使能输入%I00125 跳变为 ON 时，, 控制通讯窗口激活并设定值为 25 ms,,当使能输入跳变为 OFF 时，窗口不激活。参数块在整个存储单元%P00051。

V_I00125
MOVE UINT 1
25 IN Q V_P00051
SVC REQ
3 FNC
V_P00051 PRM
V_I00125
MOVE UINT 1
0 IN Q V_P00051

SVC_REQ 4: 改变底板通讯窗口模式和定时器值

用 SVC_REQ 4 去改变控制器通讯窗口模式和定时器值。当功能被调用，改变将在 CPU 的下一扫描期间发生

参数块有一个字长。

SVC_REQ 4 向右传递能流，除非选择的模式不是 0（限制）和 2（运行到完成）。

使底板通讯窗口失能:

使用此参数块进入 SVC_REQ 4:

地址	*高字节*	*低字节*
地址	0	0

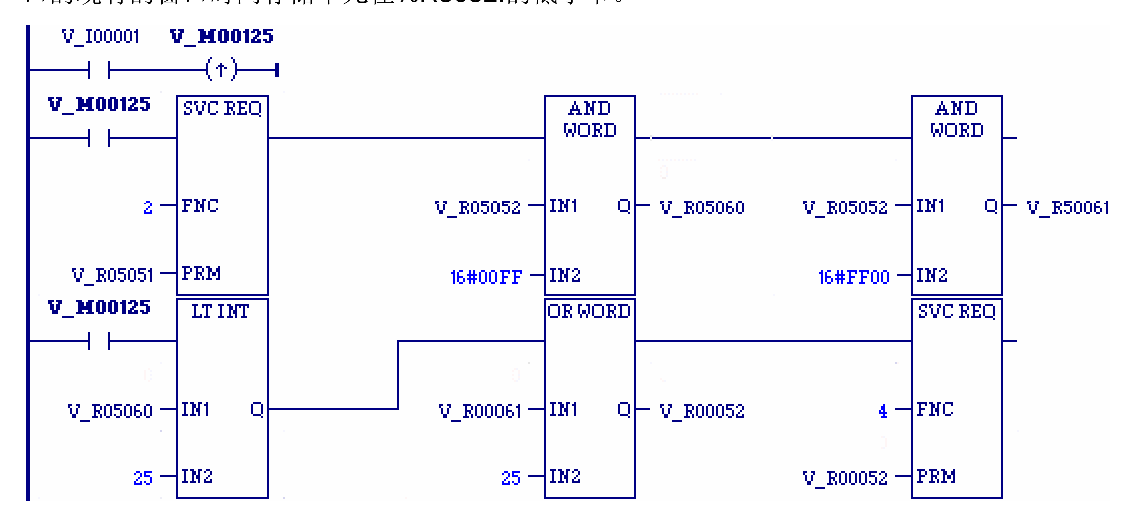
底板通讯窗口模式激活:

使用此参数块进入 SVC_REQ 4:

地址	*高字节*	*低字节*
地址	模式: 0 = 限制 2 =运行到完成	1ms ≤ 值 ≤ 255ms

例子

当使能输出%M0125 跳变为 ON 时,底板通讯窗口模式和定时器值是被读, 如果定时器值大于或等于 25 ms 时,值不会改变,. 如果小于 25 ms,值将改变为 25 ms。不管那种情况，当 梯级完成执行后窗口激活，三个窗口参数块在存储单元%R5051 中。在从读窗口值功能(SVC_REQ 2)送回参数块时，由于底板通讯窗口的模式和计时器是第二个值，底板通讯窗口的现有的窗口时间存储单元在%R5052.的低字节。



SVC_REQ 5: 改变后台任务窗口模式和定时器值

用 SVC_REQ 5 去改变后台任务窗口模式和定时器值。当功能被调用，改变将在 CPU 的下一扫描期间发生。

参数块有一个字长。

SVC_REQ 5 向右传递能流，除非选择的模式不是 0（限制）和 2（运行到完成）。

。

使后台任务窗口不激活:

使用此参数块进入 SVC_REQ 5:

地址	*高字节*	*低字节*
地址	0	0

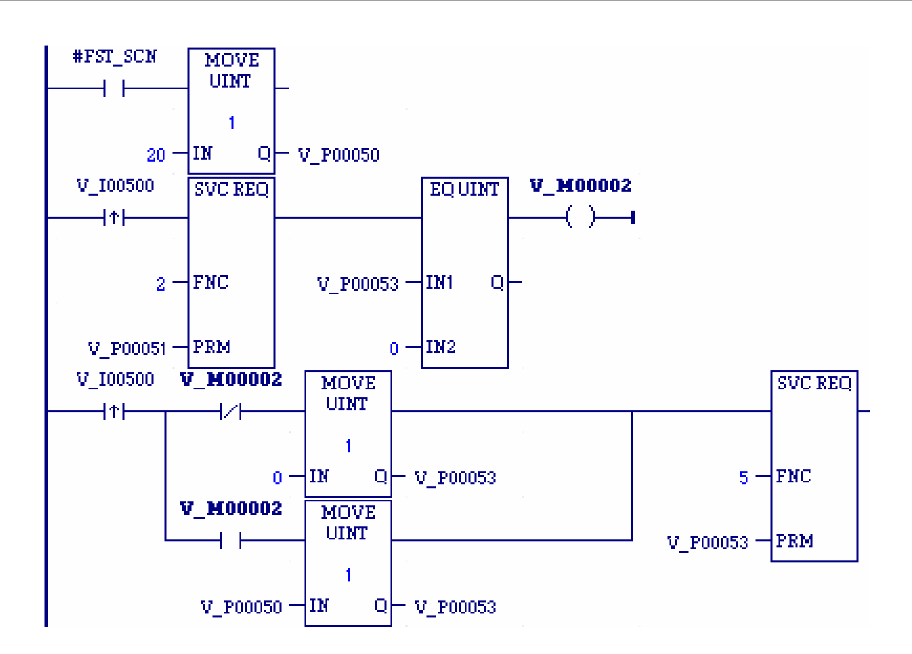
后台任务窗口模式激活:

使用此参数块进入 SVC_REQ 5::

地址	*高字节*	*低字节*
地址	模式: 0 = 限制 2 =运行到完成	1ms ≤ 值 ≤ 255ms

例子

当 使能触点#FST_SCN 第一个扫描时被置为 1，MOVE 功能用一个位于%P00050 的参数块为后台任务模块确定一个 20ms 的值.，在以后的程序中，当输入 %I00500 跳变为 ON 时，背静任务窗口的状态锁定在 ON 和 OFF，三个窗口的参数块地址在%P00051 ，时背静任务窗口时间是第三个值，这个值在从读窗口值功能(功能 #2)送回的参数块中。因此，背静窗口现有的窗口时间地址是%P00053 。


#FST_SCN
MOVE
UINT
1
20
IN
Q
V_P00050
V_I00500
SVC REQ
EQ UINT
V_M00002
2
FNC
V_P00053
IN1
Q
V_P00051
PRM
0
IN2
V_I00500
V_M00002
MOVE
UINT
1
SVC REQ
0
IN
Q
V_P00053
5
FNC
V_M00002
MOVE
UINT
V_P00053
PRM
1
V_P00050
IN
Q
V_P00053

SVC_REQ 6: 改变/读字数来求校验和

用 SVC_REQ 6 去读程序中的当前字数并求校验和或者置一个新的字数。缺省状况是校验 16 个字。除非作为被请求的操作而输入的数不是 0 或 1，那么这一功能是成功的。

这个参数块有两个字长。

读字数

在参数块的第一个字内输入零

地址	0
地址 + 1	忽略

此功能在参数块的第二个字送回当前的求校验和(字数),读功能没有指定的范围,被送回的值是当前被求校验和字的数目.

地址	0
地址+ 1	当前字计数

置一个新的字数

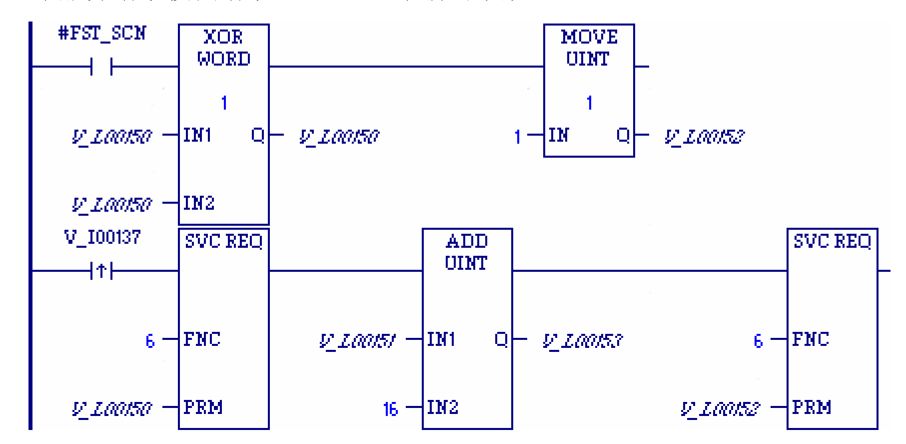
在参数块的第一个字输入一 1，在第二个字里输入新字数

地址	1
地址 + 1	新字数

CPU 把被求校验和的字数改为在参数块第二个字的值, 该值已被化为和它接近的 8 的倍数。为使求校验和不激活，置新的字数为 0 。

例子

当使能触点#FST_SCN 被置 1，用于求校验和任务功能的参数块就建立了，在后来的程序中，当输入%I00137 跳变为 ON 时，，被 求校验和的字数从 CPU 操作系统中读出，这一数以 16 为一个单位增加，并把 ADD_UINT 功能块的结果置于“hold new count for set”参数，第二个服务请求模块请求 CPU 置一个新的字数。



例中的参数块所在地址是%L00150. 他们包含以下内容:

%L00150	0 = 读当前计数
%L00151	保留当前计数
%L00152	1 = 设置当前计数
%L00153	为设置保持新计数

SVC_REQ 7: 读或改变日时间时钟

用 SVC_REQ 7 去读或改变 CPU 的日时间时钟，除了以下情况该功能有效：

- 对请求功能输入了一个无效数值
- 数据格式无效
- 提供想不到的数据格式

参数块格式

参数块的头两个字，设置为读或设为时间和日期，使用以下格式：

地址	*2 位数　年格式*	*4 位数　年格式*
地址 (字 1)	0 = 读时间和日期	0 =读时间和日期
	1 = 设置时间和日期	1 =设置时间和日期
地址+1 (字 2)	0 = 数字数据格式	80h =数字数据格式
	1 = BCD 格式	81h = BCD 格式
	2 = 未打包 BCD 格式	82h =未打包 BCD 格式
	3 = 打包 ASCII 格式(含有空格和冒号)	83h =打包 ASCII 格式
	4 = POSIX 格式	n/a
地址+2 (字 3) 到结束	数据	数据

字 3 到参数块结束包含的是通过读功能送回的输出数据，或者更改功能提供的新数据. 在这两种情况下，数据字的格式是相同的。当读日期和时间时，到参数块结束的字在输入中是被忽略的。

参数块格式和长度取决于表示年所需的位数和数据格式

数据格式和 N 位年	*参数块长度 (字数)*
BCD, 2 位　年	6
BCD, 4 位　年	6
POSIX 格式	6
未打包 BCD 2	9
未打包 BCD 4	10
数字的 (2 和 4 位数　年)	9
打包 ASCII, 2 位数　年	12
打包 ASCII, 4 位数　年	13

在任何格式中:

- 小时是以 24 小时格式存储的.
- 一周的天是从 1(星期天)到 7(星期六)的数字值

值	*星期*
1	星期天
2	星期一
3	星期二
4	星期三
5	星期四
6	星期五
7	星期六

BCD, 2 位数 年

在 BCD 格式中，每个时间和日期项占用的是一个字节,所以参数块有六个字。第六个字的最后一个字节不用.当设置时间和日期时,忽略这一字节, 当读时间和日期时,功能送回的是零字符(00)

参数块格式	*地址*	*例子(Sun., July 3, 2005, at 2:45:30 p.m. = 14:45:30 in 24-hour format)*
1 =改变， 0 = 读	地址	0 (读)
1 (BCD 格式)	地址+1	1 (BCD 模式)

高字节	*低字节*	*地址*	*高字节*	*低字节*
月	年	地址+2	07 (七月)	05 (年)
小时	天	地址+3	14 (小时)	03 (日)
秒	分钟	地址+4	30 (秒)	45 (分钟)
(空)	星期	地址+5	00	01 (星期天)

BCD, 4 位数 年

在此种格式下所有字节都使用

参数块格式	*地址*	*例 (Sun., July 3, 2005, at 2:45:30 p.m. = 14:45:30 in 24-hour format)*
1 = 改变， 0 = 读	地址	00 (读)
81h (BCD 格式, 4 位数)	地址+1	81h (BCD 格式, 4 位数)

高字节	*低字节*	*地址*	*高字节*	*低字节*
年	年	地址+2	20 (年)	05 (年)
天	月	地址+3	03 (天)	07 (七月)
分钟	小时	地址+4	45 (分钟)	14 (小时)
星期	秒	地址+5	01 (星期天)	30 (秒)

POSIX 格式

自从 1970 年 1 月 1 日午夜， POSIX 格式的时钟用的是两个带符号的 32-位的 整数 (两个 DINT) 去显示秒数和十亿分之一秒，.使用 POSIX 格式读时钟可以更容易的计算时差， 这一格式也可以用于对其他使用 POSIX 时间格式的设备进行通讯 。读/写使用 POSIX 格式的日期和时间，使用此参数块进入 SVC_REQ 7 :

参数块格式	*地址*	*例子* **December 1, 2000 at 12 noon**
1 = 写 or 0 = 读	地址	0
4 (POSIX 格式)	地址+1	4
秒 (LSW)	地址+2	975,672,000
(MSW)	地址+3	
十亿分之一秒(LSW)	地址+4	0
(MSW)	地址+5	

The PACSystems 的 CPU'的最大 POSIX 时钟值是 7FFFFFFF (十六进制 l) 秒和 999,999,999 (十进制) 纳秒，对应的是 2038 年 1 月 19 日上午 3:14 。这是 SVC_REQ 7 能够接受的 POSIX 的最大改变值，这也是一旦时钟停止后 SVC_REQ 7 将送回的最大的 POSIX 值 。

如果 SVC_REQ 7 接收一个无效的 POSIX 时间写入时钟，不会改变时钟并且使能流输出不激活。

注意: 当以 POSIX 格式读 PACSystems CPU 时钟。不容易被人眼识别，若需要，可以用应用逻辑把 POSIX 时间转换成年，月，日，小时，秒。

未打包 BCD (2 位数 年)

在未打包 BCD 格式,时间和日期的每一数字占用的时一个字节的低四位, 高四位始终为零,这一格式需九个字,值是十六进制.

参数块格式	*地址*	*例***(Thurs., Dec. 8, 2002, at 9:34:57 a.m.)**
1 =写， 0 = 读	地址	0h
2 (未打包 BCD 格式)	地址+1	2h

高字节	*低字节*		*高字节*	*低字节*
	年	地址+2	00h	02h
	月	地址+3	01h	02h
	天	地址+4	02h	08h
	小时	地址+5	00h	09h
	分钟	地址+6	03h	04h
	秒	地址+7	05h	07h
	星期	地址+8	00h	05h

未打包 BCD (4 位数 年)

在未打包 BCD 格式,时间和日期的每一数字占用的时一个字节的低四位, 高四位始终为零,这一格式需九个字,值是十六进制.

参数块格式	*地址*	***例子(Thurs., Dec. 8, 2002, at 9:34:57 a.m.)***
1 = 写， 0 = 读	地址	0h
82h (未打包 4 位数 BCD 格式)	地址+1	82h

高字节	*低字节*		*高字节*	*低字节*
	年	地址+2	00h	02h
	月	地址+3	01h	02h
	天	地址+4	00h	08h
	小时	地址+5	00h	09h
	分钟	地址+6	03h	04h
	秒	地址+7	05h	07h
	星期	地址+8	00h	05h

数字型, 2-位数 年

在数字格式,年,月,天,小时,分钟,秒和星期各自使用一个无符号整数,为使用数字格式读/写时间和日期,使用该参数块进入 SVCREQ 7:

参数块格式	*地址*	*例* ***Wed., June 15, 2005, at 12:15:30 a.m.***
1 = 写， 0 =读	地址	0
0 (数字格式, 2 位数 年)	地址+1	0

高字节	*低字节*		*值*
	年	地址+2	05
	月	地址+3	06
	天	地址+4	15
	小时	地址+5	12
	分钟	地址+6	15
	秒	地址+7	30
	星期	地址+8	04

数字型, 4 位数 年

在数字格式,年,月,天,小时,分钟,秒和星期各自使用一个无符号整数,为使用数字格式读/写时间和日期, 使用该参数块进入 SVCREQ 7:

参数块格式	*地址*	*例* ***Wed., June 15, 2005, at 12:15:30 a.m.***
1 =写， 0 =读	地址	0
80h (数字格式, 4 位数 年)	地址+1	80h

高字节	*低字节*		*值*
	年	地址+2	2005
	月	地址+3	06
	天	地址+4	15
	小时	地址+5	12
	分钟	地址+6	15
	秒	地址+7	30
	星期	地址+8	04

打包 ASCII, 位数 年

在打包 ASCII 格式, 每个时间和日期的数字都是 ASCII 格式化的字节 ，为了打印或显示，在数据中加入空格和冒号，在参数块中以 ASCII 格式表示 2 数字年需要 12 个字，而且这个值是 16 进制的 。

参数块格式	*地址*	*例* ***(Mon., Oct. 5, 2005, at 11:13:25 p.m. = 23:13:25 in 24-hour format)***
1 = 写， 0 = 读	地址	0h (读)
3 (ASCII 格式)	地址+1	3h (ASCII 格式)

高字节	*低字节*		*高字节*	*低字节*
年	年	地址+2	35h (5)	30h (0)
月	(空格)	地址+3	31h (1)	20h (空格)
(空格)	月	地址+4	20h (空格)	30h (0)
天	天	地址+5	35h (5)	30h (leading 0)
小时	(空格)	地址+6	32h (2)	20h (空格)
: (冒号)	小时	地址+7	3Ah (:)	33h (3)
分钟	分钟	地址+8	33h (3)	31h (1)
秒	: (冒号)	地址+9	32h (2)	3Ah (:)
(空格)	秒	地址+10	20h (空格)	35h (5)

星期	星期	地址+11	32h (2 = 星期一.)	30h (leading 0)

打包 ASCII, 4 位数 年

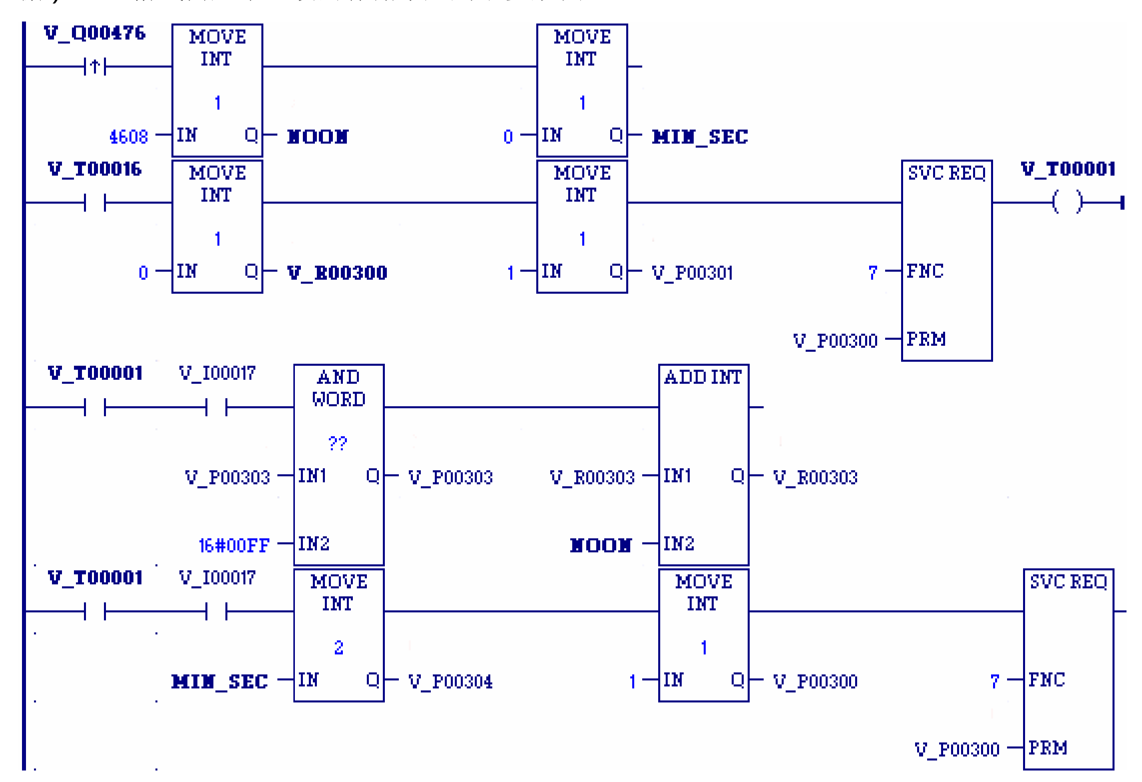
在参数块中用 ASCII 格形式表示 4 位数年需要 13 个字，而且这个值是 16 进制的 。

参数块形式	*地址*	*例* ***(Mon., Oct. 5, 2005, at 11:13:25 p.m. = 23:13:25 in 24-hour format)***
1 = 写， 0 = 读	地址	0h (读)
83 (ASCII 格式)	地址+1	83h (ASCII 该式, 4 位数)

高字节	*低字节*		*高字节*	*低字节*
年 (百)	年 (千)	地址+2	30h (0)	32h (2)
年 (个)	年 (十)	地址+3	35h (5)	30h (0)
月 (十)	(空格)	地址+4	31h (1)	20h (空格)
(空格)	月 (个)	地址+5	20h (空格)	30h (0)
天 (个)	天 （十）	地址+6	35h (5)	30h (第一位 0)
小时 (十)	(空格)	地址+7	32h (2)	20h (空格)
: (冒号)	小时 (个)	地址+8	3Ah (:)	33h (3)
分钟 (个)	分钟 (十)	地址+9	33h (3)	31h (1)
秒 (十)	: (冒号)	地址+10	32h (2)	3Ah (A)
(空格)	秒 (个)	地址+11	20 (空格)	35 (5)
星期 (个)	星期 (十)	地址+12	32h (2 = Mon.)	30h (leading 0)

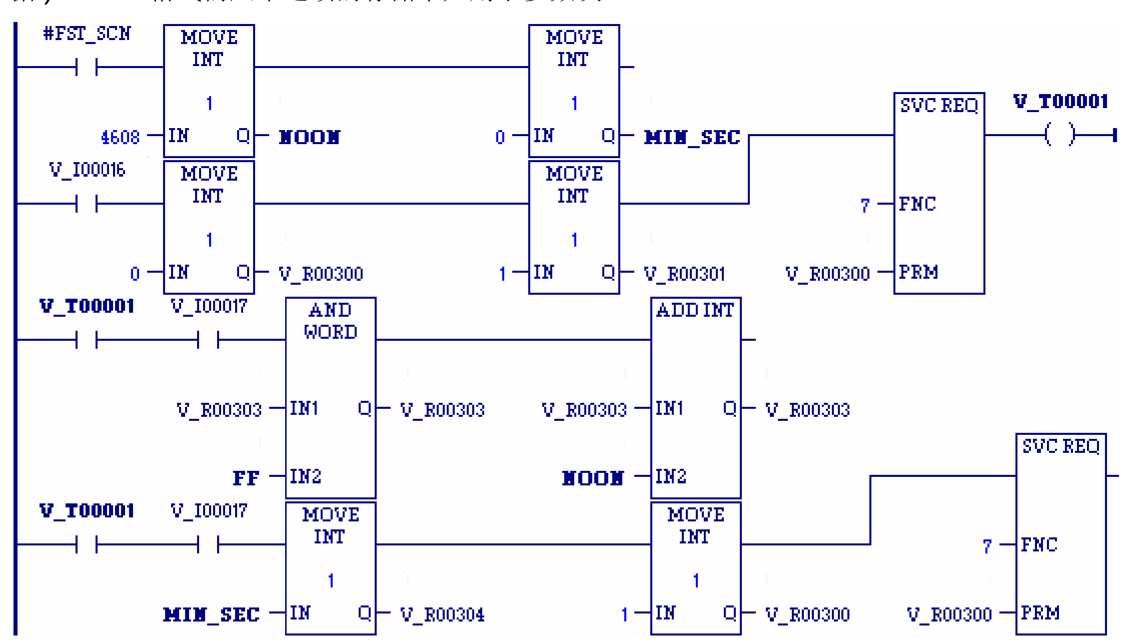
例 1 – SVC_REQ 7

当输出%Q00476 接通, 为日时间时钟建立一个参数块用于首个请求的当前日期和时间，并使用 BCD 格式设定到中午 12 点，参数块在全局数据存储单元 %P00300 里。Array NOON 已经设置在程序的其他地方，它包含 12, 0, 和 0. (Array NOON 必须包含在%R0300 里的数据.) BCD 格式需六个连续的存储单元用于参数块。



例 2 – SVC_REQ 7

当先前的逻辑有要求, 为日时间时钟建立一个参数块。参数块请求当前日期和时间，并使用 BCD 格式设置始终为中午 12 点。参数块在全局数存储单元 %P00300 里. Array NOON 已经设置在程序的其他地方，它包含 12, 0, 和 0. (Array NOON 必须包含在%R0300 里的数据.)。 BCD 格式需六个连续的存储单元用于参数块。



SVC_REQ 8: 复位看门狗定时器

在扫描过程中使用 SVC_REQ 8 去复位看门狗计时器。通常,当看门狗定时器终止时,CPU 无报警关闭。在执行长时间任务时，SVC_REQ 8 允许定时器继续工作（例如,等待来自通讯线路的响应）。

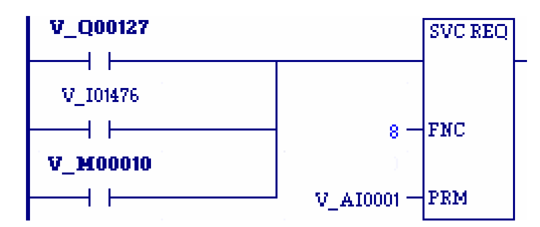


确定复位计时器时不会对控制过程产生负面影响

SVC_REQ 8 与参数块没有关系,但是, 必须指定一个 SVC_REQ 8 不用的虚拟参数。

例子

能流通过使能输出 %Q0127 或 输入量%I1476 或内部线圈%M00010 复位看门狗定时器。



SVC_REQ 9: 从开始扫描读扫描时间

用 USVC_REQ 9 读从扫描开始的时间，以毫秒为单位。这个数据格式是无符号的 16 位整数 。

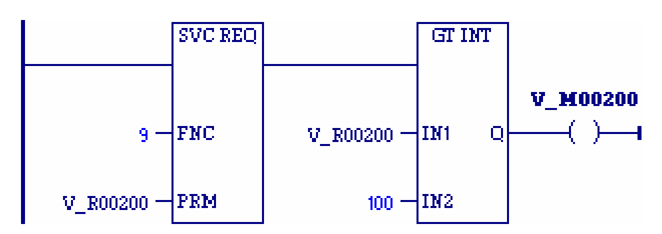
输出

这个参数块仅是一个输出参数块，它有一个字长。

地址	从开始扫描的时间

例

把从扫描开始的总时间读入%R00200. 如果它大于 100ms, 内部线圈%M0200 接通。



注意: 使用9-**错误！未定义书签。**页描述的 SVC_REQ 51，可获得更高的分辨率（十亿分之一秒）。

SVC_REQ 10:读目标名

用 SVC_REQ 10 去读当前正在执行的目标名 。

输出

输出参数块有四个字长，它送回八个 ASCII 字符: 目标名(一到七字符)后紧跟着零字符(00h). 最后一个字符总是零字符，如果目标名少于七个字符 ，最后一位补零字符。

地址	*低字节*	*高字节*
地址	字符 1	字符 2
地址+1	字符 3	字符 4
地址+2	字符 5	字符 6
地址+3	字符 7	00

例

当输入%I0301 为 OFF, 寄存器存储单元%R0099 值设为 10（它是“读目标名功能”的功能代码）, 调用程序块 READ_ID 检索目标名，参数块位于存储单元%R0100 中。

程序块 READ_ID:

SVC_REQ 11: 读控制器 ID

用 SVC_REQ 11 去读控制器执行程序名。

输出

输出参数块有四个字长，它送回八个 ASCII 字符: 目标名(一到七字符)后紧着零字符(00h). 最后一个字符总是零字符，如果目标名少于七个字符 ，最后一位补零字符添加到。

地址	*低字节*	*高字节*
地址	字符 1	字符 2
地址+1	字符 3	字符 4
地址+2	字符 5	字符 6
地址+3	字符 7	00

例

当输入%I0303 为 ON, 寄存器存储单元%R0099 值设为 11（它是读 PLC ID 功能的功能代码）, 用程序块 READ_ID 检索 ID，参数块位于地址%R0100 里。

V_I00303
MOVE WORD
1
11 IN Q V_R00099
READ_ID
CALL

程序块 READ_ID:

#ALW_ON
SVC REQ
V_R00099 FNC
V_R00100 PRM

SVC_REQ 12: 读控制器运行状态

用 SVC_REQ 12 去读 CPU 当前运行状态 。

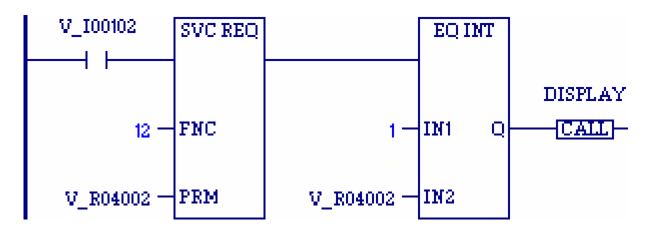
输出

这个参数块仅是一个输出参数块，它有一个字长。

地址	1 = 运行/不激活
	2 = 运行/激活

例

当 V_I00102 为 ON, CPU 运行状态读入存储单元%R4002. 如果状态是运行/不激活， CALL 功能调用程序块 DISPLAY。



SVC_REQ 13: CPU 停止

指定的扫描次数完成后，用 SVC_REQ 13 去停止 CPU 。在下一个 CPU 扫描开始时，所有的输出送回到指定的缺省状态。 一个 “PLC 停止” 故障信息放进 PLC 故障表。 I/O 按设定继续扫描 。SVC_REQ 13 有一个一个字长的输入参数块。

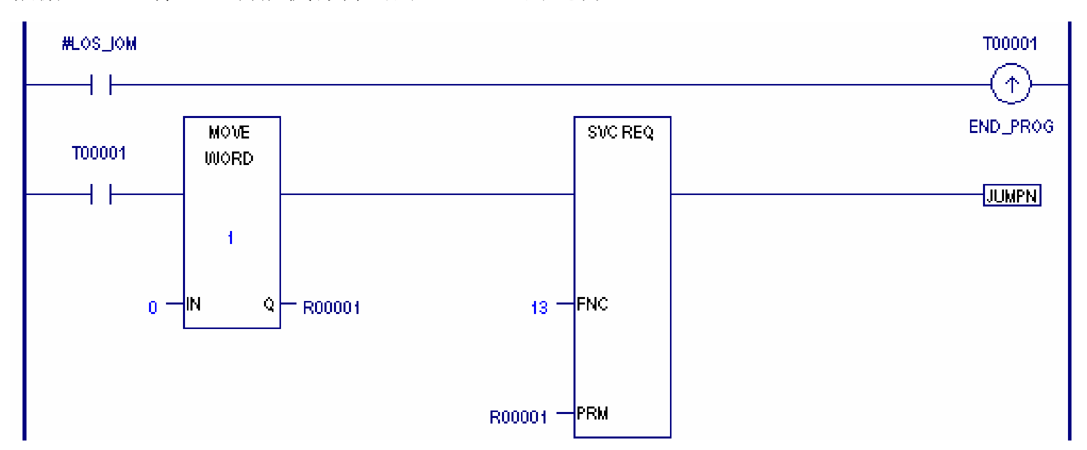
<table>
<tr><td>地址</td><td>扫描数。 有效值:
-1：CPU 使用在“硬件配置扫描”表中配置的“最后的扫描数”的值来决定什么时间跳变到停止模式。硬件配置参数的详细说明参见第 3 章。.
0 到 5： CPU 完成该扫描，然后执行这个扫描数并转到停止模式 。</td></tr>
</table>

注意: 如果 CPU 的固件版本低于 2.00, 参数块有一个字长，这个值必须置为 0; 否则 CPU 不会停止。

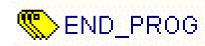
例

当发生一个“I/O 模块丢失”故障时，#LOS_IOM 触点变为 ON， SVC_REQ 13 执行.

在此例中, 如果“CPU 停止”功能成功执行，那么当前扫描中，到程序结束的 JUMP 防止根据“CPU 停止”功能执行得出的 JUMPN 的逻辑。



块的最后指令是 LABELN:



SVC_REQ 14: 清除 PLC 或 I/O 故障表

用 SVC_REQ 14 去清除 PLC 故障表或 I/O 故障表。SVC_REQ 输出置为 ON，除非输入的请求操作数不是 0 或 1 。

参数块有一个字长，并且仅是输入参数块，没有输出。

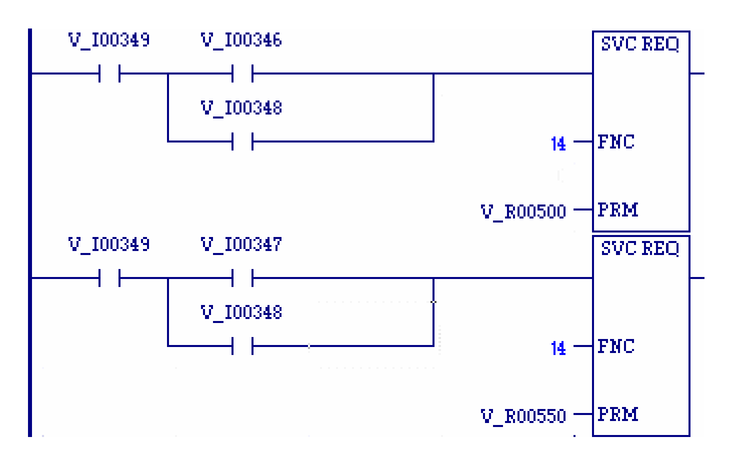
地址	0 =清除 PLC 故障表
	1 = 清除 I/O 故障表

例

当输入%I0346 和 %I0349 为 ON, PLC 故障表被清除，当输入%I0347 和 %I0349 为 ON, I/O 故障表被清除，当输入%I0348 和输入%I0349 为 ON, 两个都被清除。

PLC 故障表的参数块在%R0500 中； I/O 的故障表的参数块在%R0550 中。

注意: 两个参数块建立在程序的其他地方。



SVC_REQ 15: 读最后进入故障表的条目

Use 用 SVC_REQ 15 去读 PLC 的故障表或 I/O 故障表的最后的条目。 SVC_REQ 的能流输出为 ON，除非作为请求操作的输入值无效或故障表为空。

这个参数块有 22 个字长。第一个字仅用于输入，而其它字仅用于输出。

输入参数块

地址	*格式*
地址	0 = 读 PLC 故障表
	1 = 读 I/O 故障表
	80h = 读扩展 PLC 故障表
	81h =读扩展 I/O 故障表

输出

输出参数块的格式取决于 SVC_REQ 15 是否读 PLC 故障表、 I/O 故障表、扩展 PLC 故障表、扩展 I/O 故障表。

PLC 故障表 输出格式		***地址***	***I/O 故障表 输出格式***	
高字节	***低字节***		***高字节***	***低字节***
0		地址	1	
未用	长/短	地址+1	存储器类型	长/短
未用	未用	地址+2		偏移量
槽位	机箱号	地址+3	槽位	机箱号
	任务	地址+4	块	总线
故障动作	故障组	地址+5		点
错误代码		地址+6	故障动作	故障范畴
故障特定数据		地址+7	故障类型	故障范畴
		地址+8 到 地址+18	故障特定数据	故障描述
分钟	秒	地址+19	分钟	秒
天	小时	地址+20	天	小时
年	月	地址+21	年	月
毫秒 (*仅扩展格式*)		地址+22	毫秒(*仅扩展格式*)	
保留 (*仅扩展格式*)		地址+23	保留 (*仅扩展格式*)	

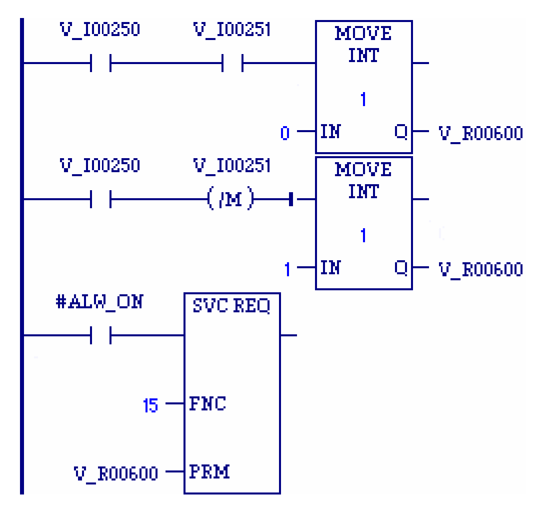
长/短 值

字 *address +1* 的第一个字节（低字节）包含的数是显示在故障条目故障特定数据的长度。可能的值有：

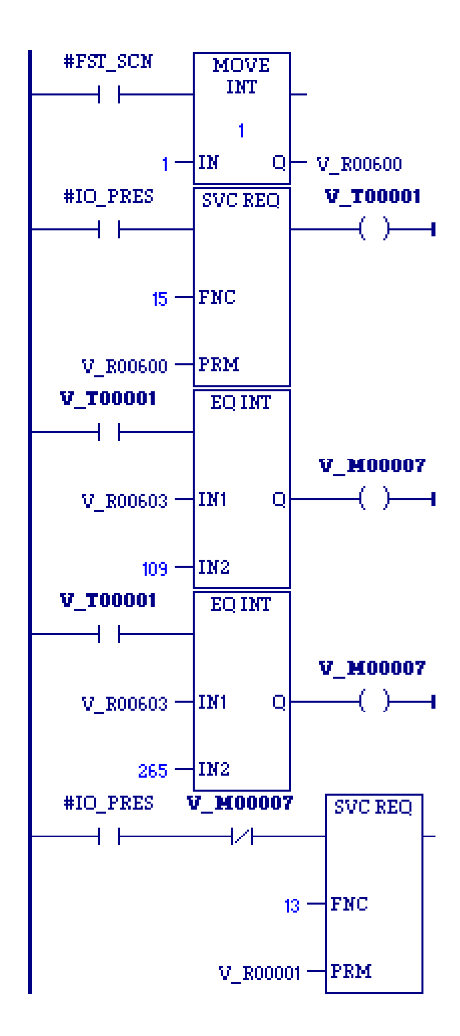
PLC 故障表	00 = 8 字节(短)
扩展 PLC 故障表	01 = 24 字节(长)
I/O 故障表	02 = 5 字节 (短)
扩展 I/O 故障表	03 = 21 字节 (长)

例 1

当输入%I0250 和%I0251 都为 ON 时, 第一个 MOVE 功能块把 0(读 PLC 故障表)放入 SVC_REQ 15 的参数块。当输入%I0250 为 ON 而输入 %I0251 为 OFF，MOVE 指令在 SVC_REQ 参数块中置 1 (读 I/O 故障表) 。参数块在存储单元%R0600 中 。



例 2



除了在 0#机箱，9 插槽和 1#机箱，9 插槽的模块发生故障，其他发生在 I/O 模块的任何故障都将使 CPU 关闭。如果故障发生在这两个模块，系统仍然运行。 表类型" 参数在第一次扫描时设置，当触点 IO_PRES 置位, 表示 I/O 故障表有一个条目。 当故障逻辑放置一个故障在故障表中之后，CPU 在扫描中置位常开触点。如果两次连续的扫描都有故障放进故障表中，那么对两个连续扫描常开触点都被置位。

本例使用一个在%R0600 中的参数块。 SVC_REQ 功能执行之后，参数块第四、五、六个字包含了故障 I/O 模块的地址。

	高字节	*低字节*
%R0600		1
%R0601		长/短
%R0602	参考地址	
%R0603	插槽数	机箱数
%R0604	块 （总线地址）	I/O 总线没有
%R0605		点地址
%R0606	故障数据	

在程序中， EQ_INT 块把故障表中的机箱/插槽地址和十六进制常数相比较，当机箱/插槽发生的故障与上述标准相同时，线圈%M0007 接通，如果%M0007 为 ON, 它的常闭触点为 OFF， 防止线圈%M0007 失电。相反的，如果 %M0007 为 OFF，因为故障发生在不同的模块，所以，常闭触点为 ON 并且线圈%M0007 失电

SVC_REQ 16:读运行累计时间时钟

用 SVC_REQ 16 去读系统运行累计时间时钟。运行累计时间时钟以秒为单位测量从 CPU 上电后的时间。 参数块有三个字长，仅用于输出。

输出

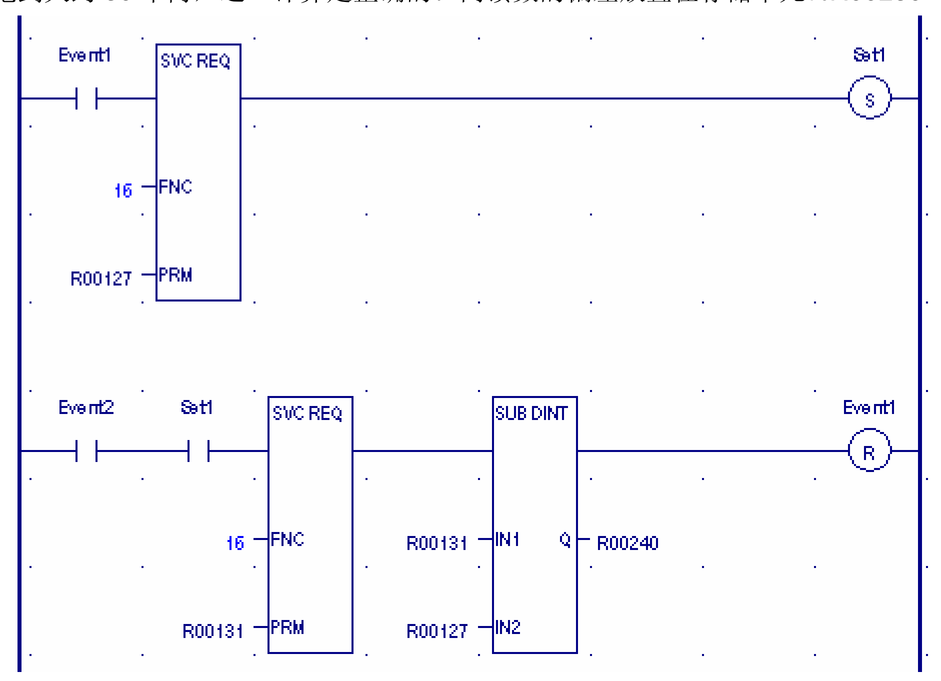
地址	上电以来时间，以秒衡量 (低位)
地址+1	上电以来时间，以秒衡量（高位)
地址+2	100 微秒记号

头两个字以秒记运行累计时间，最后一个字是当前秒的 100 微秒数。

例

当触点 Event1 关闭，带在存储单元%R00127 中的参数块的 SVC_REQ 读系统运行累计时间时钟，内部线圈 Set1 置位。当 Event2 关闭，带在存储单元%R00131 中的参数块的 SVC_REQ 读系统运行累计时间时钟。

减法功能可得到第一个和第二个读数之间的偏差，第一个和第二个读数存储在 SVC_REQ 的参数块%R00127 和%R00131 中，减法功能忽略百毫秒数， DINT 类型是有符号的值。自上电到大约 50 年内，这一计算是正确的。两读数的偏差放置在存储单元%R00250 中。



注意: 使用9-48 页描述的 SVC_REQ 50 可获得更高的分辨率（纳秒级）。

SVC_REQ 17:屏蔽/非屏蔽 I/O 中断

用 SVC_REQ 17 屏蔽或不屏蔽掩饰来自输入/输出板的中断。当一个中断被屏蔽，CPU 不执行相应的由于输入跳变引起的中断子程序。

参数块仅是一个输入参数块，有 3 个字长。

地址	0 = 非屏蔽输入 1 = 屏蔽输入
地址+1	存储器类型
地址+2	基准地址 (偏移量)

“存储器类型“是十进制数，存储在字（地址+1）的低字节，它和输入的存储器类型是一致的。

70	%I 位模式存储器
10	%AI 存储器
12	%AQ 存储器

除了以下情况其他都可以成功执行：

- 作为请求功能输入的某个数不是 0 或 1
- 被屏蔽或不屏蔽的输入/输出存储器类型不是%I, %AI 或 %AQ 存储器
- I/O 板不支持输入/输出模式
- 指定的参考地址和一个有效的中断触发参考地址不一致。
- 在配置中，指定通道没有使能信号.

屏蔽/非屏蔽模块中断

在模块配置期间，来自模块的中断信号可能是激活或不激活。如果模块的中断信号不激活，它将不能在应用程序中用于触发逻辑执行功能，它也不能被屏蔽。但是，如果在配置中模块的中断信号是激活的，在系统运行期间能够被应用程序动态地屏蔽或不屏蔽。

使用服务请求功能模块#17 可使应用程序屏蔽或不屏蔽激活的中断信号。为了屏蔽或不屏蔽从一个非 GE Fanuc VME 模块的断信号，应用逻辑应该给 SVC_REQ 17 送 VME_INT_ID(17 十进制, 11H)作为存储器类型，送 VME 的中断信号的 ID 作为偏移量, 。

当中断不被屏蔽，CPU 处理中断并确定有关的程序逻辑执行时间表。当中断被屏蔽，CPU 处理中断信号但没有确定有关的程序逻辑执行时间表。

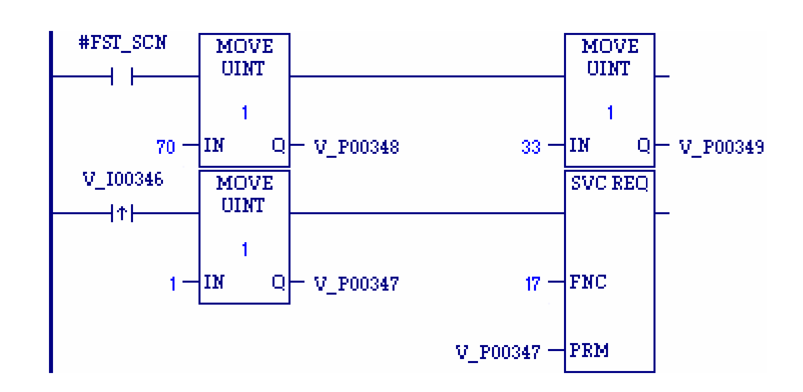
当 CPU 从停止跳变运行，中断是被屏蔽的。

关于配置和使用在 PACSystems RX7i 控制系统中 VME 模块中断的补充说明，请参阅 *PACSystems RX7i User's Guide to Integration of VME Modules,* GFK-2235。

例 1

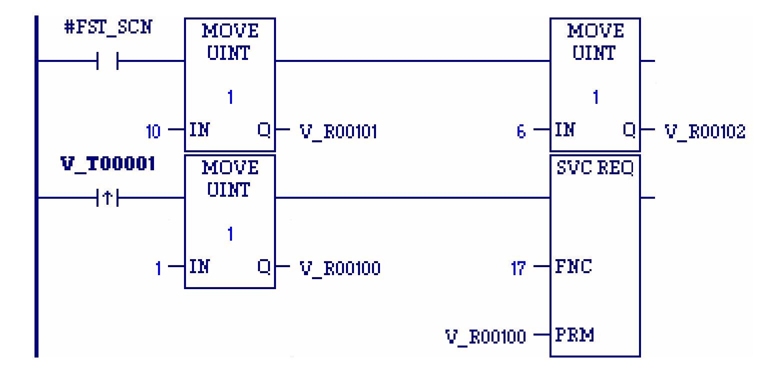
此例中，来自%I00033 的中断被屏蔽。在第一次扫描中，下列值被送进开始于%P00347 的参数块：

地址	%P00347	1	来自输入的被屏蔽的中断
地址 + 1	%P00348	70	输入类型是 %I.
地址 + 2	%P00349	33	偏移量是 33.



例 2

当%T00001 跳变为 ON, 来自%AI0006 的报警中断被屏蔽，在第一次扫描时，在%R00100 中的参数块被建立。



SVC_REQ 18:读 I/O 强制状态

用 SVC_REQ 18 去读 CPU 中的%I 和%Q 存储器区域强制值的当前状态 。

注意: SVC_REQ 18 不检测在%G 或 %M 存储器类型中覆盖。 用%S0011 (#OVR_PRE) 来检测%I, %Q, %G, %M, 和符号存储器类型中覆盖。

参数块有一个字长且仅用于输出。

输出

地址	0 = 没有置强制值
	1 =置强制值

例

SVC_REQ 把 I/O 强制值状态读入存储单元%R1003。如果在%R1003 中送回值是 1, 那么有一个强制值并且 EQ INT 使 线圈%T0001 为 ON.

SVC_REQ 19: 设置运行激活/不激活

用 SVC_REQ 19 来允许 LD 程序控制 CPU 的运行模式。.

被传递的参数表示执行了哪一个功能块。如果功能块成功执行，OK 输出变为 ON 。如果请求的操作不是 SET RUN DISABLE 模式(1)或 SET RUN ENABLE 模式(2)，那么 OK 输出置为 OFF 。

参数块仅是输入参数块，有以下格式：

地址	1 = SET RUN DISABLE 模式 2 = SET RUN ENABLE 模式

例

当输入%I00157 跳变到 ON，设置为 RUN DISABLE 模式。当 SVC_REQ 功能块成功执行，线圈%Q00157 变为 ON。当 %Q00157 变为 ON 并且寄存器 %R00099 大于零, 模式变为 RUN ENABLE 模式。当 SVCREQ 成功执行线圈%Q00157 变为 OFF。

V_I00157 MOVE UINT 1 IN Q V_R00100 SVC REQ FNC PRM 19 V_R00100 V_Q00157 (S)

V_Q00157 GT UINT V_R00099 IN1 Q 0 IN2 MOVE UINT 1 2 IN Q V_R00100 SVC REQ 19 FNC V_R00100 PRM V_Q00158 (R)

SVC_REQ 20: 读故障表

用 SVC_REQ 20 去检索整个 PLC 或 I/O 故障表并把它送回到指定寄存器的 LD 程序中。

第一个输入参数指定去读哪一个表，第二个输入参数（对于一个标准的“读故障表”总是 0）用于扩展格式，读指定的故障条目或一定范围的故障条目。故障表数据放在紧随输入参数的参数块中。

如果功能执行成功，那么 OK 输出变为 ON。如果请求操作不是读 PLC 故障表 (00h), 读 I/O 故障表 (01h), 读扩展 PLC 故障表 (80h), 或读扩展 I/O 故障表 (81h), 那么 OK 输出为 OFF，否则没有足够的存储器来存储故障数据。如果指定故障表是空的，功能设置输出 OK 为 ON，但是只送回故障表的标题信息 。

这一参数块是输入和输出参数块，它有两种格式：

- 无扩展(读 PLC 故障表 (00h), 读 I/O 故障表(01h))
- 扩展 (读扩展 PLC 故障表(80h), 或读扩展 I/O 故障表 (81h))

无扩展格式

对于无扩展格式 SVC_REQ 20，需 693 个寄存器可用。

输入参数块格式

地址	00h = 读 PLC 故障表 01h = 读 I/O 故障表
地址+1	总为 0 (0)

输出参数块格式

地址	0 = PLC 故障表 1 = I/O 故障表
地址+1	总为 0 (0)
地址+2	被读的故障数
地址+3 到 地址+14	未用
地址+15 地址+16 地址+17	自从最后一次清除的时间
地址+18	自从最后一次清除的故障数
地址+19	队列的故障数
地址+20	读故障数
地址+21 到地址+36	CPU 名
地址+37	故障数据开始

在无扩展格式, 每个故障表条目长 21 字长(42 个字节). 最多有 16 PLC 故障表条目和 32 I/O 故障表条目。如果故障表条目读为空，那么没有数据被送回。

下表显示的是 PLC 故障表条目 和 I/O 故障表条目的送回格式。

故障表条目的送回格式

地址	***PLC 故障表条目***	***I/O 故障表条目***
地址**+38**	长/短	长/短
地址**+39**	未用	给定地址
地址**+40** 地址**+41**	PLC 故障地址	I/O 故障地址
地址**+42**	故障组和作用	I/O 故障地址
地址**+43**	出错代码	故障组和作用
地址**+44**	故障附加数据	故障范畴和类型
地址**+45**	故障附加数据	故障描述
地址**+46 到**地址**+55**	故障附加数据	故障特殊数据
地址**+56 到**地址**+58**	时间标记	时间标记

（地址+38）的第一个字节中 的长/短指示器定义为送进故障条目中的故障数据的数量。在 PLC 故障表中， 一个长/短值 00 表示送进故障条目中故障附加数据的 8 个字节， 01 表示故障附加数据的 24 个字节。在 I/O 故障表，02 表示故障特殊数据的 5 个字节，03 表示 21 个字节。

每个字段的解释，参阅 12 章, “故障手册.”

扩展格式

对于扩展格式(读扩展 PLC 故障表 (80h), 或读扩展 I/O 故障表 (81h)), PLC 计算被读的条目数并当没有足够的存储器时送回一个错误信息。

对于扩展故障格式故障表的总大小是

标题尺寸 + ((# 故障条目) * (故障条目尺寸))

输入参数块格式

地址	80h = 读扩展 PLC 故障表 81h = 读扩展 I/O 故障表
地址+1	被读故障开始地址
地址+2	被读故障数

输出参数块格式

地址	80h = 扩展 PLC 故障表 81h = 扩展 I/O 故障表
地址+1	被读故障开始地址
地址+2	被读故障数
地址+3 到地址+14	未用
地址+15 **地址+16** **地址+17**	自从最后一次清除以来的时间
地址+18	自从最后一次清除以来的故障数
地址+19	队列中的故障数
地址+20	读故障数
地址+21 到地址+37	CPU 名
地址+38 到地址+58	故障数据 – 同无扩展格式
地址+59	时间标记 (毫秒)
地址+60	未用

对于读扩展 PLC 故障表 (80h) 和读扩展 I/O 故障表(81h)，每个扩展故障表条目有 23 个字长 (46 字节).。最多有 40 个 PLC 故障表条目和 40 个 I/O 故障表条目。PLC 故障表条目和 I/O 故障表条目的缺省值分别为 16 和 32 。如果被读的故障表是空，那么没有数据送回。

SVC_REQ 20 举例

例 1:无扩展模式

当输入 %M00033 跳变为 ON， 读 PLC 故障表 ， 。当 %M00034 跳变为 ON, 读 I/O 故障表.。参数块在存储单元%R00500 中。 当 SVC_REQ 功能成功执行, 线圈 %M00035 接通。

例 2: 扩展模式

当输入%M00033 跳变为 ON, 读扩展 PLC 故障表，参数块在存储单元%R00500 中。%R00500 包含故障表类型(PLC 扩展)；%R00501 包含读到的开始故障，%R005002 包含读到的以在%R00501 里的故障编号开始的故障数量。当 SVC_REQ 功能成功执行, 线圈%M00034 接通。

V_M00033
MOVE WORD 1
80 IN Q V_R00500
MOVE WORD 1
1 IN Q V_Q00501
MOVE WORD 1
10 IN Q V_R00502
SVC REQ
V_M00034 (S)
20 FNC
V_R00500 PRM

SVC_REQ 21: 自定义的故障记录

用 SVC_REQ 21 定义一个能在 PLC 故障表中显示的故障。这个故障含有二进制信息或 ASCII 码信息。自定义故障代码从 0 hex 开始。

对于一个被显示的"Application Msg:"来说，故障的错误代码信息必须在 0 到 2047 之间。如果错误代码是在十进制的 81 到 112 范围内， CPU 将对%SA 系统存储器相同编号的故障位置位。允许有 32 个位分别被置位。

错误代码	*状态位*
错误 0—80	无位被置
错误 81—112	%SA 被置
错误 113—2047	无位被置
错误 2048—32,767	保留

当 EN 激活, 由 IN 定位的故障数据组被作为一个故障记录进 PLC 故障表。如果 IN 失不激活，OK 位清零。如果错误代码超出范围，OK 位被清零，故障不会被记录。

这一参数块仅是输入参数块。格式如下：

参数地址	*错误代码*	
	MSB	***LSB***
地址+1	正文 2	正文 1
地址+2	正文 4	正文 3
地址+3	正文 6	正文 5
地址+4	正文 8	正文 7
地址+5	正文 10	正文 9
地址+6	正文 12	正文 11
地址+7	正文 14	正文 13
地址+8	正文 16	正文 15
地址+9	正文 18	正文 17
地址+10	正文 20	正文 19
地址+11	正文 22	正文 21
地址+12	正文 24	正文 23

输入参数数据允许在 0 到 2047 之间选择一个错误代码，正文信息放在一个长 PLC 故障的故障附加数据部分。PLC 故障地址，故障组和故障动作通过功能模块输入 。

故障正文字节 1-24 可以用来传递故障的二进制或 ASCII 数据。如果故障正文数据的第一位不为零，那么数据将是一个 ASCII 信息串，这一信息将在故障表的故障描述区显示。如果

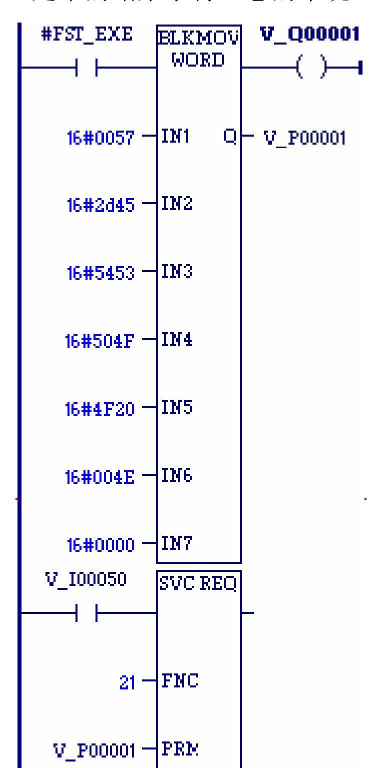
信息少于 24 个字符，那么 ASCII 串必须是无效字节终止。 程序装置将显示“Application Msg:”，在“Application Msg:”后面的 ASCII 数据被作为信息立即显示出来。 如果错误代码在 1 和 2047 之间, 错误代码号在“Application Msg:”串后后立刻被显示. (如果错误代码大于 2047，该功能被忽略且无能流送回)

如果正文的第一个字节为零， 那么在故障描述中仅显示 “Application Msg:” 。用户数据记录中剩下的 1-23 字节被认为是二进制数据，这个显示数据显示在 PLC 故障表中 。

注意: 当自定义故障显示在 PLC 故障表中，显示的是错误代码加上-32768 (8000 hex)。例如，错误代码 5 显示为-32763.。

例

送给 IN1 的是故障错误代码 。送进的值是 16x0057，错误代码用十进制表示就是 87，这个值作为故障信息的一部分显示。下一输入值给出错误信息正文的 ASCII 代码。对 IN2, 输入 2D45. 低字节, 45,解码为字母 **E** ， 高字节, 2D, 解码为 **-**. 用这种方式继续下去，将得到 **S T O P** 和 **O N**， 最后字符是 **00**， 是串的结束字符。总的来说，译码产生串信息 **E_STOP ON**.



SVC_REQ 22: 屏蔽/非屏蔽定时中断

用 SVC_REQ 22 来屏蔽或不屏蔽定时中断并读当前屏蔽 。当中断被屏蔽，CPU 不会执行任何定时中断块、与定时中断块有关定时程序。定时中断作为组被屏蔽或不屏蔽 。他们不能单独地被屏蔽或不屏蔽。

除非作为请求操作或屏蔽而输入值不是 0 或 1，否则该功能将成功执行。这一参数块是输入和输出参数块。

用以下格式决定当前屏蔽：

地址	0 = 读中断信号屏蔽

CPU 送回格式：

地址	0 =读中断信号屏蔽
地址+1	0 =定时中断不被屏蔽 1 =定时中断被屏蔽

改变当前屏蔽用以下格式；

地址	1 = 屏蔽/不屏蔽中断
地址+1	0 =不屏蔽定时中断 1 =屏蔽定时中断

例子

当输入%I00055 跳变 为 on,定时中断被屏蔽。

V_I00055
MOVE UINT
1
1 IN Q V_R01002
MOVE UINT
1
1 IN Q V_R01003
SVC REQ
22 FNC
V_R01002 PRM

SVC_REQ 23: 读主求校验和

用 SVC_REQ 23 来读用户程序和配置组的主求校验和，并读取组成该服务请求的块的求校验和。

该服务请求没有输入参数块. 输出参数块需要 15 个字存储空间。

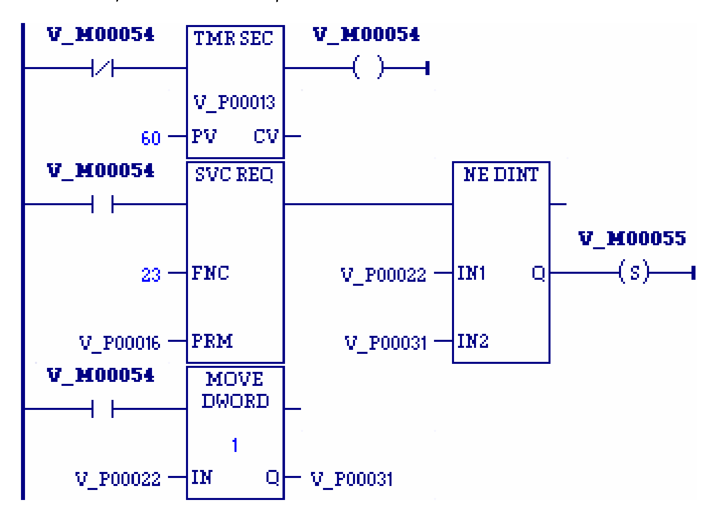
输出

当一个 RUN MODE STORE 激活，在存储完成之前，程序求校验和无效。. 为确定求校验和什么时间有效，在输出参数块的开始提供了三个标记（分别表示 Program Block Checksum, Master Program Checksum, Master Configuration Checksum）。

地址	描述
地址	程序 求校验和有效(0 = 无效, 1 = 有效)
地址+ 1	主程序求校验和 有效 (0 = 无效, 1 = 有效)
地址+ 2	主配置求校验和 有效 (0 = 无效, 1 = 有效)
地址+ 3	LD/SFC 块数 (包括 _MAIN 块)
地址+ 4	用户程序大小，一字节为单位。 (DWORD 数据类型)
地址+ 6	程序设置附加求校验和
地址+ 7	程序 CRC 求校验和 (DWORD 数据类型)
地址+ 9	配置数据的大小， 以千字节为单位
地址+ 10	配置附加求校验和
地址+ 11	配置 CRC 求校验和 (DWORD 数据类型)
地址+ 13	高字节：总为零 低字节：当前执行块的附加求校验和
地址+ 14	当前执行块的 CRC 求校验和

例子 – SVC_REQ 23

当使用寄存器%P00013 到 %P00015 的定时器终止, 执行读求校验和读。 求校验和数据送回到寄存器%P00030 到%P00016 中。在寄存器 %P00022 和%P00023(程序求校验和是一个 DWORD 数据类型并占用两个相邻的寄存器)中主程序求校验和与最后存入的主程序求校验和相比较。如果它们不同，那么线圈 %M00055 锁在 ON 。然后 当前主程序求校验和被保存在寄存器%P00031 和 %P00032 中。



SVC_REQ 24: 模块复位

用 SVC_REQ 24 去复位子板或一些模块。支持 SVC_REQ 24 的模块包括：

RX3i IC693BEM331, IC694BEM331, IC693APU300, IC694APU300, IC695ETM001, IC693ALG2222, IC694ALG2222

RX7i: 嵌入以太网接口模式, IC697BEM731, IC698BEM731, IC697HSC700, IC697ALG230, IC698ETM001

除非下列情况存在，否则 SVCREQ 输出置为 ON：

-输入一个无效的插槽或机箱数

-指定区域没有模块

-指定区域模块不支持运行时间复位

-CPU 不能复位指定区域模块

这一功能,参数块有一个字长，它仅是一个输入参数块

地址	模块插槽(低字节) 模块机箱 (高字节) *机箱 0, 插槽 1 表示一个复位送到子板*

注意: 每次在一个扫描周期中为一个给定模块调用 SVC_REQ #24 是很重要的。每次执行这一功能, 目标模块将被复位，不管它是否完成了先前的复位。

当发送了一个 SVC_REQ #24 到模块后, 必须等待 5 秒钟才能发送另一个 SVC_REQ #24 到同一模块 。要确定模块有时间重新恢复和完成它的启动。

例

该例复位的是在 0 机箱/2 插槽的模块。在第 1 梯级中，当触点 %I00200 闭合, 每次扫描正跳变线圈置%I00250 为 ON 。在第二梯级的 MOVE_WORD 指令中接收能流 并把值 2 传送进 %R00500. 。在第三梯级的 SVC_REQ 功能块接收能流并复位在%R00500 中以机箱/插槽值表示的模块 。

SVC_REQ 25：激活/不激活 EXE 块和独立 C 程序求校验和

用 SVC_REQ 25 来激活或不激活在后台 求校验和计算中包括 EXE。 这一缺省包括求校验和.

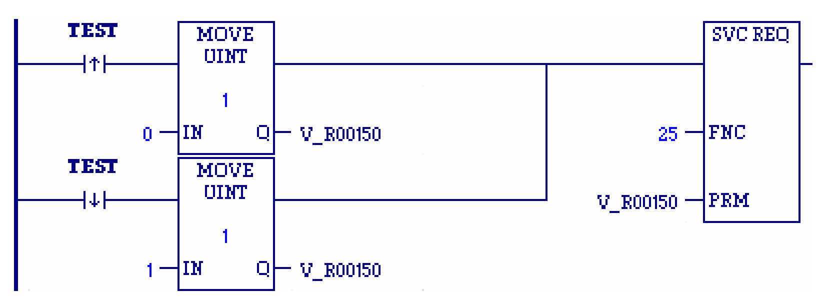
这一服务请求使用的仅是一个输入参数块.

地址	0 = 在求校验和计算中不包括 C 应用
	1 =在求校验和计算中包括 C 应用程序包

参数块在执行完服务请求后不会改变。

例

当线圈 TEST 由 OFF 跳变为 ON, SVC_REQ 25 执行在后台 求校验和计算中不包括 EXE 程序。当线圈 TEST 从 ON 跳变为 OFF, SVC_REQ 重新执行在后台 求校验和计算中包括 EXE 块 。



SVC_REQ 29:读失电累计时间

用 SVCREQ 29 来读最后失电和最近得电之间的时间总和. 如果看门狗定时器在失电之前终止，PLC 不能计算失电累计时间，所以这个时间设置为零。

这一服务请求不能从一个 C 块被访问。

这一功能有一个仅是输出的参数块，它有三个字长。

地址	失电累计时间（秒）(低位)
地址+ 1	失电累计时间（秒）(高位)
地址 + 2	100μS 记号

头两个字是失电流累计时间的秒数，最后一个字当前秒的 100μS 数。

注意: **尽管该要求响应的分辨率是** 100μS, 但实际度为 1 秒。当 PLC 失电时后备电池时钟精确到 1 秒以内。

SVCREQ 29 的子例

该例中当输入%I0251 为 ON,累计失电时间放置在以%R0050 起始的参数块中。输出线圈(%Q0001)变为 on。



SVC_REQ 32:暂停/恢复 I/O 中断

使用 Use SVC_REQ 32 来暂停一组 I/O 中断,导致中断排队等待直到中断恢复. 可以排队的 I/O 中断数量可以根据 I/O 模块的容量决定。CPU 发出命令决定 I/O 模块中断是被暂停或恢复,I/O 模块的默认状态是恢复,暂停将应用于与 I/O 模块相关的所有 I/O 中断,中断在一个扫描周期里被暂停或恢复. SVC_REQ 32 只使用一个输入参数块,长度为 3 个字.

地址	0 = 恢复中断 1 = 暂停中断
地址 + 1	存储器类型
地址+ 2	参考 (偏移量)

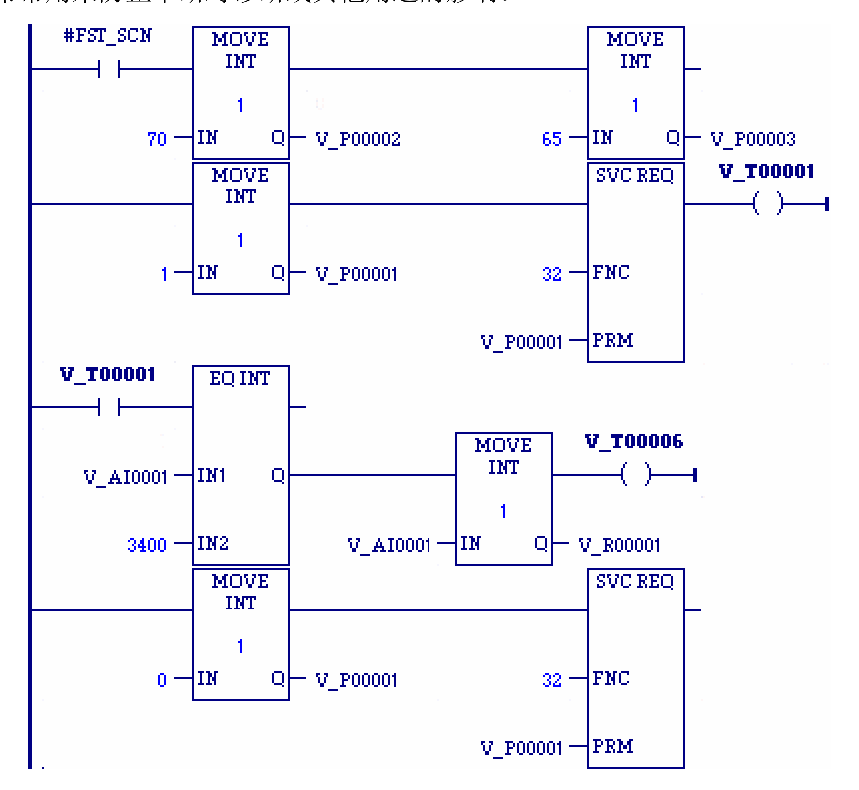
以下情况将不被执行:

- 不同于 0 或 1 的数字作为第一个参数.
- 存储器类型参数不是 70(%I 存储器)
- 与指定地址相关的模块不是一个恰当的模块
- 指定的参考地址不是高速计数器的第一个%I 参考值.
- CPU 和 I/O 模块之间的通讯失败(摸板不存在或有一个致命的错误)

SVC_REQ 3 2 的例子

当 CPU 执行了第二行的逻辑关系时,来自起始点地址为%I00065 的高速计数器模块的中断将被暂停.如果不暂停,当%R000001 有一个不同于 3,400 的值时,在执行第三行的过程中，将产生一个来自高速计数器模块的中断，%T00006 被置为 1。(%AI00001 是高速计数器第一个模拟量输入参考).

注意: 除非被暂停或屏蔽，否则 I/O 中断就能中断一个功能块的执行。这个服务请求功能常常用来防止中断对诊断或其他用途的影响。



SVC_REQ 45:跳过下一个扫描

使用 SVCREQ 功能 #45 来跳过下一个输出和输入扫描,。在扫描期间（在此期间执行 SVCREQ #45），任何输出参考表的改变不受相应模块的物理输出影响。在扫描期间（其后的一个扫描中执行 SVCREQ #45），模块上任何物理输入数据的改变不影响相应的输入参考表。

.这个功能没有参数块

注意: 这个服务请求由 90-30 系列应用软件的转化而来。所有 PACSystems 固件版本支持的暂停 I/O 功能块应当用于新的应用软件。

注意: 使用 SVCREQ #45 不会影响 DOIO 功能块.。象 SVCREQ #45 一样在同一逻辑程序中使用 DOIO 将更新 I/O。

例

下例中，当“Idle”触点传递能流时，下一个输入和输出扫描将被跳过。

IDLE
SVC REQ
45 FNC
R00001 PRM

SVC_REQ 50:读累计时间时钟

用 SVC_REQ 50 去读系统的累计时间时钟。累计时间时钟以秒为单位测量 CPU 自上电以来的时间。

这一参数块有四个字长且仅用作输出。

输出

地址	从上电开始的秒数(低位)
地址+1	从上电开始的秒数(高位)
地址+2	十亿分之一秒标记(低位)
地址+3	十亿分之一秒标记(高位)

头两个字是以秒为单位的计累计时间。第二个两个字是当前秒累计的十亿分之一秒数的数量。

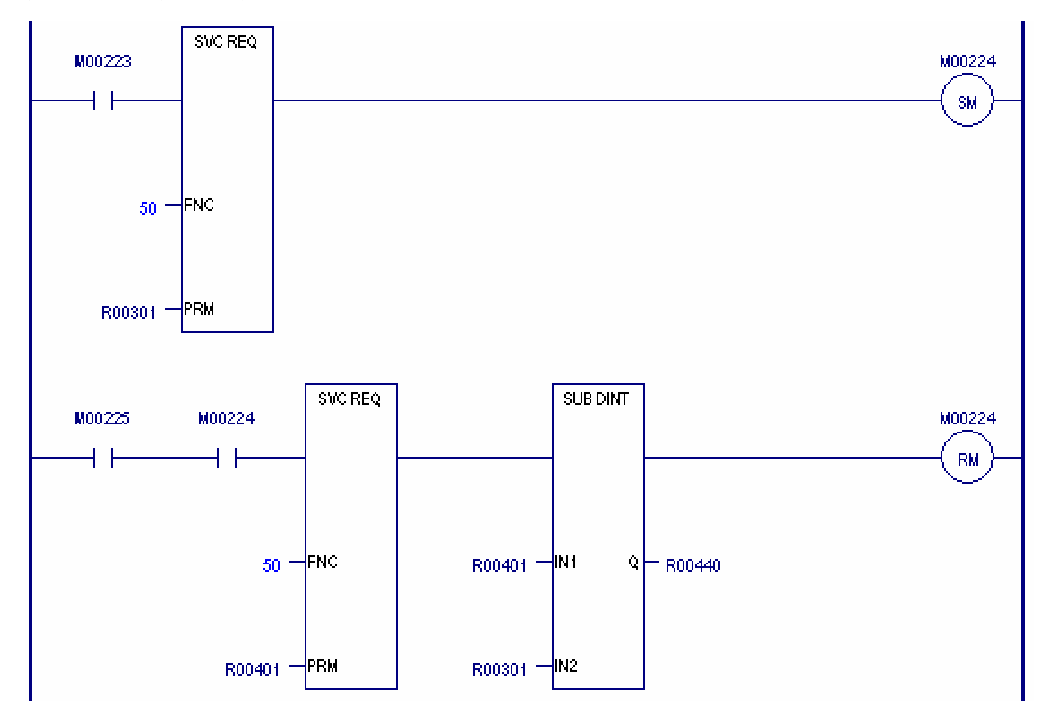
SVC_REQ 5 0 的例子

当内部触点%M00223 为 on, 带在%R00301 中参数块的 SVC_REQ 读系统累计时间时钟，置内部线圈%M00224 为 1。

当%M00224 和%M00225 为 on, 带在%R00401 中参数块的 SVC_REQ 再一次读累计时间时钟。

减法功能得到第一次和第二次读数的差值，并存储在 SVC_REQ 参数块%R00401 和%R00301 中。减法功能忽略 DINT 实际是一个有符号值的事实。 自从上电开始这一计算能保持大约 50 年的正确性，例中忽略了十亿分之一秒字段。

两次读数的差值放置在%R00440.



SVC_REQ 51:从开始扫描读扫描时间

用 SVC_REQ 51 以十亿分之一秒为单位读从开始扫描的 时间。 数据是无符号 32 位整数。

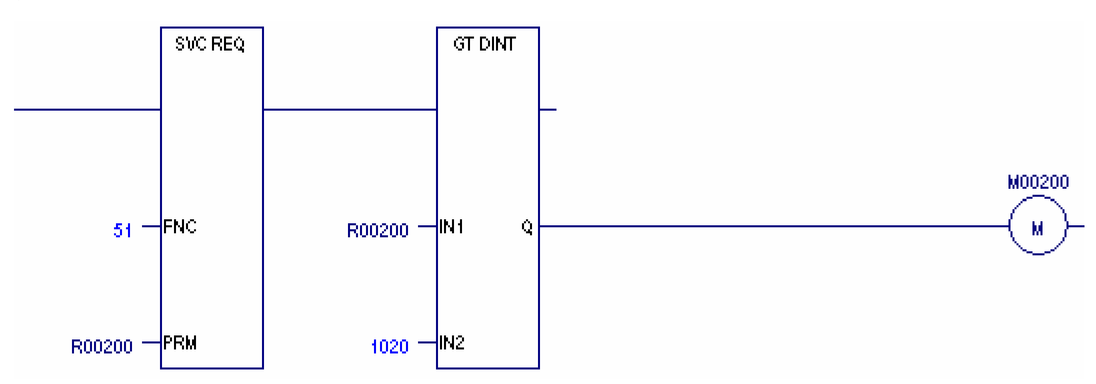
输出

参数块仅是输出参数块，它有两个字长。

地址	从开始扫描的时间(单位：十亿分之一秒) – 低位
address+1	从开始扫描的时间(单位：十亿分之一秒) – 高位

例

从扫描开始的累计时间被读到存储单元%R00200 和%R00201 中，如果它大于 10,020ns, 内部线圈%M0200 变为 ON。



Chapter 10 PID 函数

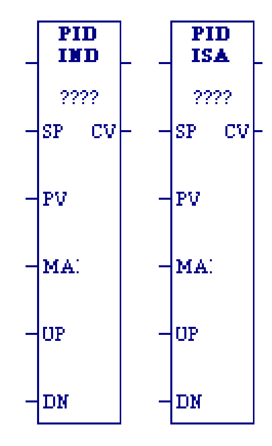
本章描述 PID 函数。PID 函数用于闭环过程控制。PID 函数将当前值(PV)与设定值(SP)进行比较，并根据偏差更新控制变量(CV)

- PID 函数的格式
- PID 函数的操作
- PID 函数的参数块
- PID 函数的运算法则的选择
- 确定过程特性
- 设定包括回路增益在内的参数
- 例子

PID 函数的格式

PID 函数的回路增益和其他参数存储在 40 个字(每个字 16 位)队列中，PID 函数可以通过这 40 个字在期望的时间间隔内完成 PID 运算。所有参数是 16 位整型字。

可配置参数出错时 PID 函数不执行。修改数据时，可以通过临时线圈监控。



PID 函数的操作数

参数	*描述*	*允许的操作数*	*是否可选择*
地址 (Address) (????)	PID 控制信息的存放位置(用户或内部参数)，10-5 页详述。使用 40 个不可共享的字。	R, L, P, W 和符号变量	No
设定值(SP)	控制环或过程设定值。用于过程变量计算，PID 函数调节输出控制变量以使过程变量等于设定值(0 偏差)	除了 S, SA, SB SC 和流	No
过程变量 (PV)	过程变量来自受控的过程，一般为%AI 输入	除了 S, SA, SB ， SC，常数 和流	No
手动(MAN)	如果设为 1(通过结点)，PID 块为手动调节模式。如果设为 off，PID 块为自动调节模式	流	No
UP	如果和 MAN 一起设为 1，没调用一次 PID 函数，控制变量值加 1*	流	No
DN	如果和 MAN 一起设为 1，没调用一次 PID 函数，控制变量值(CV)减 1*	流	No
控制变量 (CV)	控制变量输出到过程，通常为%AQ 输出	除了%S，常数 和流	No

* 每次访问 PID 函数加(UP 参数)1 或减 (DN 参数)1

函数的操作

自动操作

给电流到 Enable 并且不给电流到 Manual 时 PID 函数执行。PID 块将当前的 CPU 逝去时间与存储在内部变量数列的时间相比较。如果差值大于第三个字(%Ref+2)内存储的采样时间值，PID 进行运算。将上一次运行时间和 CV 输出更新。自动模式下，CV 输出存放在手动命令参数%Ref+13 中

手动操作

将电流同时给到 Enable 和 Manual 时，输出值从手动命令参数%Ref+13 设定。如果有电流给到 UP 或者 DN 输入，则手动命令在 PID 函数每次运行时给 CV 值加 1 或者减 1。要使输出 CV 变化更快，可以在手动命令字内增加/减少每次的累加值。

PID 块用 CV 上限和下限限制 CV 输出。如果设定最小变化时间，则可以 CV 输出的变化速率。如果 CV 幅度和变化率超过限幅值，存储在积分器中的数值会将 CV 调回限定范围内。这种震荡特性意味着即使偏差想要使 CV 长时间超过上/下限幅值，在偏差值变号时，CV 输出也会离开限幅值。

手动命令在自动模式下跟踪 CV 输出值，在手动模式下设定 CV 值。这种操作使自动和手动方式可以无冲击的转换。手动模式下，CV 上下限幅和最小变化时间仍然作用于 CV 输出，并且积分器存储的值被更新。这意味着，如果你在手动模式下使用手动命令，CV 输出值的改变速率不会比最小时间变化率限定的改变速率更快，也不或超过上下限幅。

PID 函数的时间间隔

PID 的执行速率最快为 10 毫秒执行一次。如果设定为每个扫描周期执行一次，并且扫描周期小于 10 毫秒，则不管扫描周期是多少，PID 都会在逝去时间累计达到 10 毫秒时才会执行。例如：如果扫描周期是 9 毫秒，则 PID 函数个一个周期执行一次，每两次执行的实际时间间隔为 18 毫秒。PID 函数不能在一个扫描周期内被多次调用。

执行 PID 函数的最长时间间隔为 10.9 分钟。PID 函数在最后一次执行后 100 毫秒内补偿实际的逝去时间。

CPU 逝去时间大于或等于最后一次执行的时间+采样时间时，执行 PID 运算。如果采样时间设为 0，函数每次被使能时即执行；然而，如上所述，PID 函数的最小执行时间为 10 毫秒

输入和输出标度转换

为了与 16 位模拟过程变量兼容，PID 函数的所有参数是 16 位整型字。一些参数需要定义为过程变量或者控制变量。

设定值必须与过程变量缩放到相同的范围内，因为这两个值的差将作为 PID 函数的偏差值。过程变量和控制变量不用做同样的缩放。设为-32000~32000 或 0~32000 来与模拟量匹配，或者设为 0~10000 来与 0.00% ~100.00%匹配。如果过程变量和控制变量没有作相同的缩放，缩放因素会记入 PID 增益中。

PID 函数的控制块

PID 函数的参数块占据了 40 个字的的存储区域。只有前 13 个字是可配置的。其余 27 个字供 CPU 使用，不可配置。每个 PID 函数调用必须使用不同的 40 个字的存储区域，即使所有的 13 个可配置变量都相同，也不允许两个 PID 函数使用存储区域。

控制块的前 13 个字必须在 PID 函数执行前设定数值。大多数的缺省值为 0。一旦选择了合适的 PID 值，可以将其作为常数在 BLKMOV 内定义，以便于程序修改。

变量序列参数

PID 函数读取 13 个参数，并将其余 27 个字用于数据存储。一般不要更改这 27 个字的值。如果在经过长时间的延时后调用自动模式的 PID 函数，你可能需要使用 SVC_REQ 16 或 SVC_REQ 51 将当前的 CPU 逝去时间写入%Ref+23 来更新 PID 的最后一次运行时间以避免积分器出错。如果将覆盖 (Override)控制字(%Ref+14)的低位设为 1,则控制字的下面 4 位必须设定为控制 PID 块输入结点，并且必须设定 SP 和 PV 值，因为你已经替代梯形图逻辑对 PID 块进行控制了。

	参数/描述	*低位单元*	*范围*
地址	**环号** PID 块的可选数字。CPU 内一个一般的标识，由用户口接设备定义	整型	0 ~ 255 (只用于显示)
地址 +1	**运算法则** 1 = ISA 运算法则 2 = 独立运算法则	-	由 CPU 设定
地址+2	**采样时间** PID 运算的最短间隔时间，增加值最小为 10 毫秒。例如，100 毫秒采样周期设定为 10	10ms	0 (每个扫描周期) ~ 65535 (10.9 分) 最小为 10 毫秒.
地址+3 地址+4	**死区 +和死区–** 整型数定义死去的上(+)下(-)限幅。如果不需要设定死区，这些数值必须为 0。如果 PID 偏差(SP - PV)或 (PV - SP)大于(-)值小于(+)值，PID 计算认为偏差值为 0。如果死区值不为 0，则(+)值必须大于 0，(-)值必须小于 0，否则 PID 块不工作。 在调节 PID 增益之前，将这些值设为 0。死区能够避免由于偏差值的微小变化而引起的输出波动。	PV 计算	死区+: 0 ~32767 (不允许为负值) 死区-: -32768 to 0 (不允许为正值)
地址+5	**比例增益-Kp** **(ISA 版本的控制器增益, Kc)** 偏差量(Error)变化 1，控制变量 CV 变化值为 Kp*1/100，即如果 Kp=450，输出量变化值为 450*Error/100。Kp 是 PID 调节的第一个增益设定。	0.01 CV%/PV%	0~327.67%
地址+6	**微分增益-Kd** 10 毫秒内偏差值 ERRO 或当前值 PV 变化 1 时，控制变量 CV 的变化量。输入值为时间，低位表示 10 毫秒。例如，Kd 输入值为 120 时表示实际的时间为 1.2 秒。即，如果偏差值 ERRO 每 30 毫秒变化 4 个当前值 PV 单位，则输出值变化量为 Kd * delta Error/delta time 或 120*4/3。Kd 用于提高慢速环的响应速度，但是对 PV 输入的噪声特别敏感。可以使用微分过滤器降低噪声干扰，可以设定配置字的第五位来实现滤波(见 10-7 页)	0.01 秒	0~327.67 秒

	参数/描述	低位单元	范围
地址+7	**积分比率-Ki** 在 ERROR 值改变一个 PV 计数值时，CV 的变化量。显示为每秒变化量。显示为 0.000 累加值/秒，Ki=1400 时显示为 1.400 累加值/秒。例如，Ki 值为 1400，error 为 20，CPU 扫描时间 50 毫秒时，输出值变化量为 1400 * 20 * 50/1000(采样周期为 0)。Ki 通常为设定了 Kp 之后第一个要设定的参数	累加值/1000 秒	0 to 32.767 累加值/秒
地址+8	**CV 误差 /输出偏移** 在到达 PID 限幅或比率前 CV 计数值的增加数。可以在使用了比例增益的情况下设定非 0 的 CV 值，或者用于从其他环控制的本环前馈控制。	CV Counts	-32768 to 32767 (增加到积分器输出上)
地址+9 地址+10	**CV 上下限幅** 定义 CV 输出的最大/最小值。这两个值必需设定。上限幅值必须是大于下限幅值的正数，否则 PID 块不工作。这两个值通常基于 CV 输出自身的值来设定，但是也可以用于调节棒图设定。CV 值达到限幅时，程序块反向修改积分器数值。	CV Counts	-32768 to 32767 (>%Ref+10)
地址+11	**最小回转时间** CV 输出从 0 到 100% 或 32000 的最小秒数。这个参数限制 CV 输出值的变化速率。 如果是正值，则在最小回转时间内，CV 变化量不能超过 32000。例如，如果采样时间为 2.5 秒，最小回转时间为 500 秒，则在 PID 的每个运算过程中，CV 值的变化量不能超过 32000*2.5/500=16。如果超过了 CV 变化率限幅，会自动调整积分器数值。如果最小回转时间设为 0，则 CV 值变化率没有限制。在调试时或者调解 PID 回环增益时，将最小回转时间设为 0。	Second/Full Travel	0 (没有限制) to 32000 秒改变 32 CV

<table>
<tr><th></th><th>参数/描述</th><th>低位单元</th><th>范围</th></tr>
<tr><td>地址+12</td><td>配置字
这个字的低 6 位用于修改 3 种标准 PID 设置。其他位应设为 0。
Bit 0: ERROR 项。这一位设为 0，则 error =SP - PV.，设为 1 则 error = PV – SP。将这位设为 1，就将 PID Error 项从正常的(SP – PV)改为(PV – SP)，更改了反馈的正负。在 PV 增大，CV 就必须减小的情况下使用这种设置。
Bit 1:输出极性。此位置 0，CV 输出表示 PID 计算的输出。此位置 1 则 CV 输出与 PID 计算的输出值由相反的极性。
Bit 2: PV 的微分。设为 0 则 对 error 项做微分，设为 1 则对 PV 进行微分。
Bit 3:死区动作。死区动作位设为 0，则没有死区。如果偏差在死区内，则此偏差必须为 0。否则，偏差不受死区影响。
如果将死区动作位设为 1，则表示使用死区。如果偏差值在死区内，则强制将偏差值设为 0。如果偏差在死区外，则偏差值减去死区限幅值。(error = error – 死区限幅)
Bit 4:反向设定动作。如果将此位设为 0，反向设定动作会使用反向计算。输出达到限幅值时，会重置累加的 Y 余数值以使输出值正确。
此位置 1 时，则在开始计算时累加 Y 项。这时，只要有输出限幅，则保持限幅值。
Bit 5:启用微分过滤器。此位置 0，不对微分项使用微分过滤。
此位置 1，应用 1 维过滤。这可以限制作用在微分项上的高频干扰。
注意: .这些位以 2 的幂次来设定。例如，要设为缺省的 PID 配置，将配置字设为 0，加 1 将 Error 项从 SP–PV 改变为 PV–SP，或者加 2 将 CV = PID Output 改变为 CV = -PID Output，或者加 4 将对 Error 变化率微分改变为对 PV 变化率微分等等</td><td>使用低 6 位</td><td>0~ 2 位用于 Error+/-, Out Polarity, Deriv.</td></tr>
<tr><td>地址+13</td><td>手动命令
PID 块采用自动模式时为当前的 CV 输出值。转换为手动模式时，用于在上下限幅和回转时间限制范围内设定 CV 输出值和积分器的内部值。</td><td>CV Counts</td><td>自动模式下跟踪 CV 值
手动模式下设定 CV 值</td></tr>
<tr><td>地址+14</td><td>控制字
如果覆盖位设为 1，控制字和内部的 SP, PV 和 CV 参数用于 PID 块的远程操作。这种方式允许远端操作接口设备 (如计算机)替代 PLC 程序进行控制。
注意:如果不想对 PID 块进行远端操作，确定控制字被设为 0。如果低位设为 0，在 PID 块运行时可以利用下面 4 位观测 PID 的状态。
前五位位置的离散数据格式如下：
<table>
<tr><th>位:</th><th>字值:</th><th>功能:</th><th>覆盖位设为 1 时的外部动作</th></tr>
<tr><td>0</td><td>1</td><td>Override</td><td>为 0，在下面监控结点状态，为 1，从外部设定。</td></tr>
<tr><td>1</td><td>2</td><td>Manual /Auto</td><td>为 1，手动模式；其他数，自动模式</td></tr>
<tr><td>2</td><td>4</td><td>Enable</td><td>一般为 1，否则不调用这个块</td></tr>
<tr><td>3</td><td>8</td><td>UP /Raise</td><td>Manual(Bit 1)和这一位同时为 1，每次计算时增加</td></tr>
<tr><td>4</td><td>16</td><td>DN /Lower</td><td>Manual(Bit 1)和这一位同时为 1，每次计算时减少</td></tr>
</table></td><td>由 CPU 保持，除非第 1 位被设定</td><td>不设定的话，由 CPU 保持否则:如果为 1，低位设定覆盖(Override)</td></tr>
</table>

	参数/描述	*低位单元*	*范围*
地址+15	**内部的 SP** 跟踪 SP 输入。 如果 Override = 1，从外部设定 SP 值(被覆盖前，保持原始的 SP 值)	由 CPU 设定和保持	不可配置
地址+16	**内部 CV** 跟踪 CV 输出		
地址+17	**内部 PV** 跟踪 PV 输入。 如果 Override = 1，必须从外部设定 PV 值		
地址+18	**输出** 用带符号字表示转换前函数块输出。如果没有配置输出转换并且控制字中的输出极性位设为 0，这个值等于 CV 输出值。如果选择转换，并且输出极性位设为 1，这个值等于 CV 输出值的相反数。		
地址+19	**微分项存储** 不要向这个位置写入数据		
地址+20 地址+21	**积分项存储** 用于存储内部积分值。不要向这个位置写入数据		
地址+22	**回转项存储** 不要向这个位置写入数据		

	参数/描述	*低位单元*	*范围*
地址+23 ~ 地址+25	**时钟** 存储内部逝去时间(PID 最后一次执行的时间)。一般不允许向这些位置写入数据。特殊情况下可以写入。*		
地址+26	**Y 余数存储** 为积分器保存余数，做 0 稳态误差调节		
地址+27 地址+28	**SP, PV 上下范围** 为 PV 计数值设定高低限幅，整型数。(Address +27 内的数值必须小于 Address +28 内的数值)	PV Counts	-32768~32767
地址+29	**保留**		
地址+30 ~地址 31	**上一次微分项存储** 用于微分过滤器计算		
地址+32 ~地址+39	**保留** 32-34 位内部使用保留; 35-39 位外部使用保留。Do not use these references.不要使用这些变量	N/A	不可配置

- PID 函数读取 13 个参数，并将其余 27 个字用于数据存储。一般不要更改这 27 个字的值。如果在经过长时间的延时后调用自动模式的 PID 函数，你可能需要使用 SVC_REQ 16 或 SVC_REQ 51 将当前的 CPU 逝去时间写入%Ref+23 来更新 PID 的最后一次运行时间以避免积分器出错。如果将覆盖 (Override)控制字(%Ref+14)的低位设为 1,则控制字的下面 4 位必须设定为控制 PID 块输入结点，并且必须设定 SP 和 PV 值，因为你已经替代梯形图逻辑对 PID 块进行控制了。

运算规则选择(PIDISA 或者 PIDIND)与增益

PID 块可以选用独立(PID_IND)运算规则，或者选用标准 ISA (PID_ISA)版本运算规则。

一般这两种 PID 类型下都有 Error= SP – PV，你可以将 Error 项(配置字 %Ref+12 的第 0 位)设为 1，这种情况下 Error= PV –SP

如果你想要 CV 输出值与 PV 输入值做反向动作 (CV 下降时 PV 上升)，而不是像正常的那样 PV 上升，CV 就上升的话，可以使用反转动作模式。

Error = (SP - PV) 或者(PV - SP)如果配置字的低位设为 1

微分一般是基于从 PID 最后一次运算到现在的 Error 值的变化的，在改变 SP 值的情况下，微分可能导致输出值产生很大的变化。如果不希望有这样的变化，可以将配置字的第三位设为 1 来计算 PV 值变化量的微分。dt 值等于当前逝去时间减去 PID 上一次执行时的逝去时间。

dt = CPU 当前逝去时间 - PID 上一次执行时的 CPU 逝去时间

微分 = (Error – 前一个 Error)/dt

或者 (PV – 前一个 PV)/dt 如果配置字的第三位设为 1

使用独立的 PID (PID_IND) 运算规则时，输出值如下:

PID 输出 = Kp * Error +   Ki * Error * dt + Kd * 微分 + CV 偏差

标准 ISA (PID_ISA) 运算规则运算公式如下:

PID 输出= Kc * (Error +   Error * dt/Ti + Td * 微分) + CV 偏差

Kc 为控制器增益，Ti 微积分时间，Td 为微分时间。ISA 的优点在于调节 Kc 时，同时改变比例，积分和微分项，这可能使调节更简单。如果你已经有 PID 增益值或者已经有 Ti 和 Td，使用

Kp = Kc   Ki = Kc/Ti   and   Kd = Kc/Td

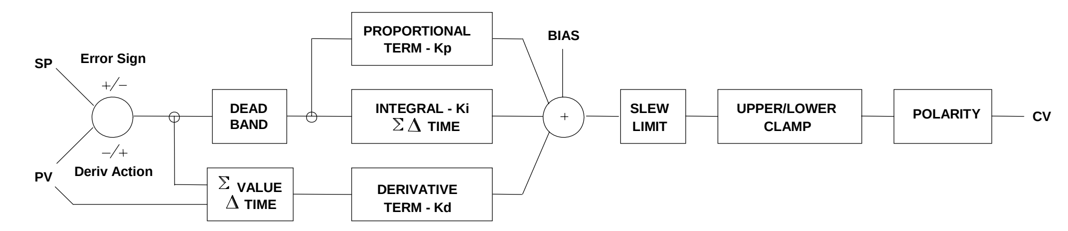
将他们转换为 PID 用户参数输入

CV 偏差项是 PID 各个部分之外的附加项。如果只使用比例项，并且 PV=SP,ERROR=0，但是又不想令 CV 等于 0 时，你可能需要使用这一项。这种情况下，当 PV=SP 时，为 CV 设定一个期望的偏差值。在前馈控制和其他 PID 环控制这个 PID 的 CV 输出的情况下，也可能用到 CV 偏差值。

如果用到积分增益 Ki，因为积分器作为可以看作自动的偏差量，所以这是一般将 CV 偏差值设为 0。启动手动模式并且使用手动命令字(%Ref+13)将积分器设定到期望的 CV，然后转到自动模式。如果将 Ki 为 0，这也起作用，除非在进入自动模式后，积分器不是基于 Error 调节的。

独立 PID 运算法则 (PIDIND)

下表显示了 PID 运算法则如何工作：



ISA (PIDISA)运算与此类似，只是 Kp 参数提到外面，积分增益= Kp * Ki，微分增益= Kp * Kd。在用户参数配置字中设定具体的位即可确定 Error 的符号，微分作用和极性。

CV 振幅和比率限制

PID 块不会直接将计算得到的 PID 输出值赋给 CV。PID 的两种运算规则都会将振幅和变化率限制强制加在输出控制变量上。最大变化率等于 CV 最大值(32767)除以最小回转时间(如果最小回转时间大于 0)。例如，如果最小回转时间为 100 秒，转换率为每秒最大 320 个 CV 计数值。如果 dt 计算时间为 50 毫秒，那麽每次改变的最大变化量为 320*50/1000 或者说 16 个 CV 计数值。

之后 CV 输出与上下限幅值比较。如果超过某个限幅值，则 CV 输出值设定为该限幅值。如果超过允许的比率或振幅，会调整内部积分器值来匹配限定值以避免 PID 重启终结

1.最后，PID 块会检查输出的极性(配置字%Ref+12 的第 2 位)，如果此位置 1，改变输出值的符号。

CV = PID 正限幅输出 或者
- PID 负限幅输出 如果输出极性位被设定

如果 PID 块为自动模式，最后的 CV 值存放在手动命令字%Ref+13 中。如果 PID 块为手动模式,CV 值由手动命令设定，但是所有的比率和限幅值仍然起作用。这意味着，手动命令不能令 CV 值大于正限幅值，小于正限幅值，也不能令 CV 变化率大于最小回转时间设定的比率。

采样时间和 PID 块时序

PID 块是数字化的模拟量控制函数，所以 PID 输出方程式中的 dt 采样时间不能设为无穷小。受控的大部分过程可用一次或二次方程近似表示，相当于纯时间滞后。PID 块设定一个 CV 输出给过程，并使用过程反馈回来的 PV 值来确定 Error，以便调节下一次的 CV 输出值。总计时间是一个关键的过程参数，这个参数确定 PV 在多长时间后响应 CV 的变化。如下面关于设定回环增益部分所讨论的，在 CV 发生阶跃时，总计时间常量，Tp+Tc，是一阶系统 PV 值达到 CV 值 63%所需要的时间。如果 PID 块的采样时间不小于总计时间的 1/2，那麽 PID 块无法实现对此过程的控制。太大的采样时间会使控制不稳定。

采样时将不应大于总计时间的 1/10(最差的情况下也不能大于总计时间的 1/5)。例如，如果预计 PV 值会用 2 秒钟的时间到达最终值的 2/3，采样时间应小于 0.2 秒，最差的情况下也不能大于 0.4 秒。另一方面，采样时间也不能太小，例如小于总计时间的 1/1000，或者积分器的 Ki * Error * dt 值趋向于 0 的情况。例如，如果有这样一个非常缓慢的过程，预计 PV 值会用 10 小时或者 36000 秒钟的时间到达最终值的 63%时，采样时间应大于或等于 40 秒。

除非过程非常快，否则不需要将采样时间设为 0，从而每个扫描周期都进行 PID 运算。如果很多 PID 回路使用了大于扫描周期的采样时间，那麼在有很多 PID 环同时完成计算时，CPU 的扫描时间会有很大的变化。解决的办法是将控制 PID 块运行的位排在一个数列内，之后按顺序的把数列内的一位或几位数据置 0。

确定过程特性

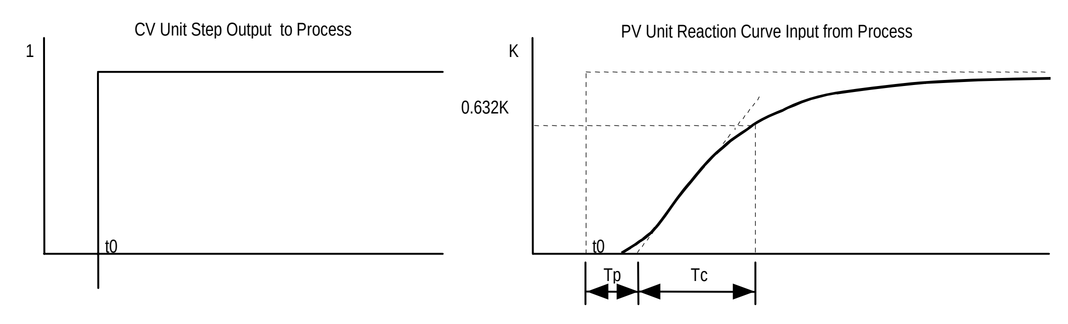
PID 回路增益 Kp, Ki 和 Kd 根据受控过程的特性确定。设定 PID 回路时有 2 个关键问题：

1. 在 CV 值发生一定量的变化时，PV 值应改变多少，或者什麼是开环增益？
2. 系统响应有多块，或者说 CV 发生阶跃时，PV 以多快的速度变化？

许多过程可以近似的用过程增益，一阶或二阶延时和纯滞后来表示。在频域内，带一阶延时的纯滞后系统的传递函数为：

PV(s)/CV(s) = G(s) = K * e **(-Tp s)/(1 + Tc s)

时域内 t0 时刻的开环单位阶跃响应曲线如下：



以下过程模式参数从 PV 的单位阶跃响应曲线得到：

K	过程开环增益=PV 最终变化量/t0 时刻 CV 变化量
Tp	t0 时刻后，PV 开始变化前的过程或通道延时或死区时间
Tc	一阶过程时间常数，Tp 之后，PV 到达最终值 63.2%之前的时间

通常测试这些参数的最快的方法是将 PID 块置于手动模式，并且给 CV 输出一个小的阶跃(改变手动命令%Ref+13)，然后测出 PV 的时域响应。对于慢速过程，可以手动完成；但是对于快速过程，则需要用数据表记录器或在计算机上记录数据曲线。CV 的步长要能够引起可观察的 PV 变化，但也不能大到引起受测过程混乱。最好的步长是 CV 上下限幅差值的 2%~10%。

包括调节回环增益在内的参数设定

因为 PID 参数依赖于受控过程，因此没办法提前设定值；然而，通常能比较简便的找到合适的回环增益：

1. 将所有用户参数设为 0，然后将 CV 上下限幅值设为允许的最大最小 CV 值。将采样时间设为估计的过程时间的 1/100~1/10。
2. 将 PID 块置于手动模式下并且将手动命令字(%Ref+13)设为不同的值，以确定 CV 是否能变化到上下限幅。记录某些 CV 点的 PV 值并且将其作为 SP 的设定值
3. 设定一个小的增益，例如 100 * 最大 CV 值/最大 PV 值，给 Kp 并且切掉手动模式。令设定值 SP 做阶跃，阶跃量为 PV 最大范围的 2% ~10%，然后观察 PV 的响应。如果响应太慢，增加 Kp 值；如果超调和振荡太大，则减小 Kp 值。
4. 确定 Kp 之后，增加 Ki 使响应在 2~3 个循环内从超调达到稳态。这可能要降低 Kp。也要尝试使用不同的步长和 CV 操作点。
5. 确定了合适的 Kp 和 Ki 增益后，在不引起震荡的前提下增加 Kd 值。一般不需要使用 Kd，如果 PV 值噪声太大，也不能使用 Kd。
6. 检查不同 SP 操作点时的增益，需要的话，增加死区和最小回转时间。一些反转操作过程可能需要设定配置字的 error 符号或者极性位

使用 Ziegle 和 Nichols 调试方法设定回环增益

一旦确定了 3 个模型参数 K, Tp 和 Tc，可以用他们评估初始的 PID 回环增益。下列方法对于振幅比率为 1/4 的干扰有很好的响应。振幅比率是闭环响应时的第二个波峰值与第一个波峰值的比值。

1. 计算反应速率：
   R = K/Tc
2. 只使用比例控制时，Kp 计算如下：()
   Kp = 1/(R * Tp) = Tc/(K * Tp)

   使用比例积分控制时：
   Kp = 0.9/(R * Tp) = 0.9 * Tc/(K * Tp) Ki = 0.3 * Kp/Tp

   使用比例，积分和微分控制时：
   Kp = G/(R * Tp) where G is from 1.2 to 2.0
   Ki = 0.5 * Kp/Tp
   Kd = 0.5 * Kp * Tp
3. 检查采样时间是否在下面的范围内：
   (Tp + Tc)/10~(Tp + Tc)/1000

Ideal Tuning Method 理想调节方法

.理想调节过程提供了对 SP 变化的最好的响应，只对 Tp 过程延时或对死区时间延时

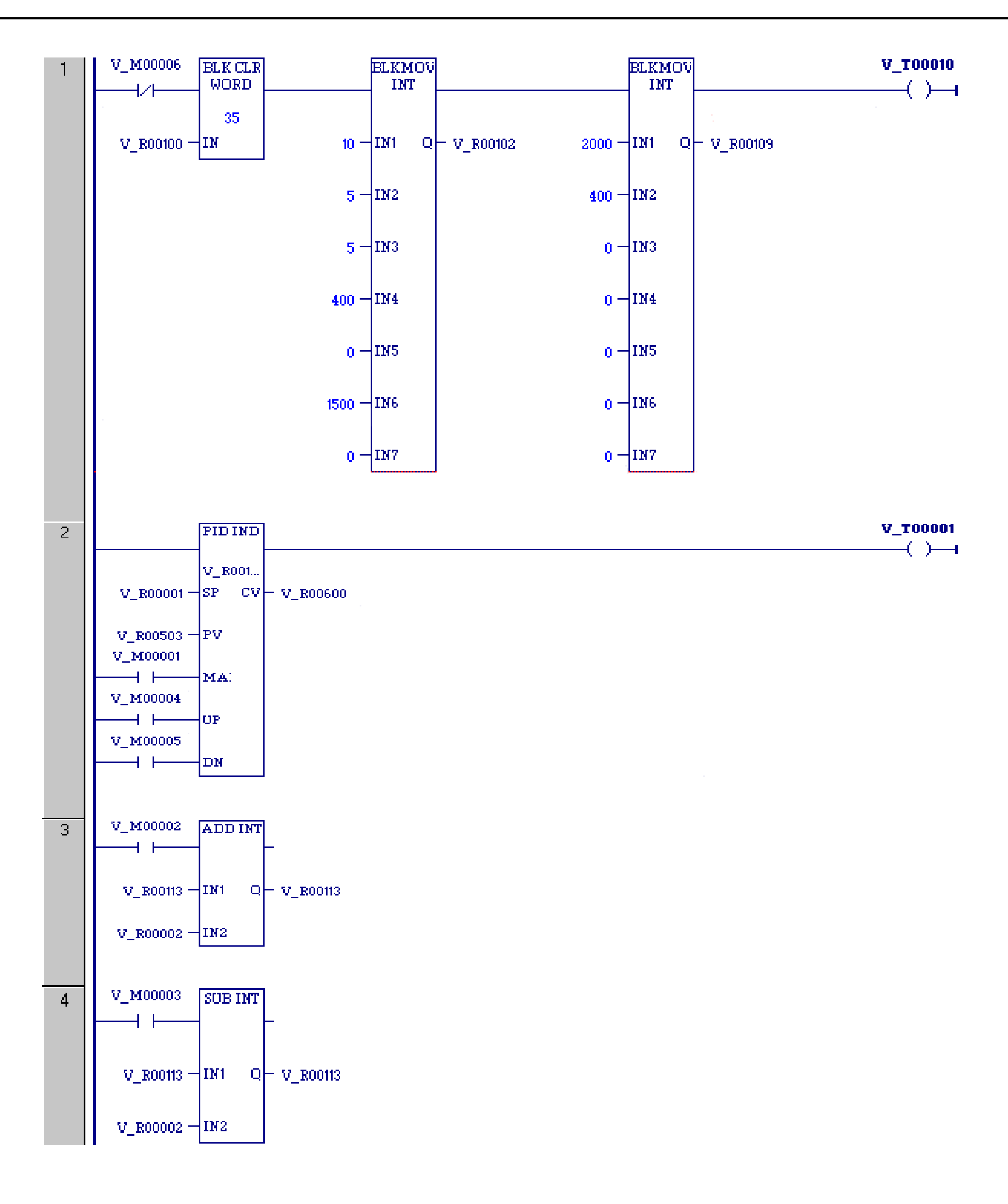
Kp = 2 * Tc/(3 * K * Tp)
Ki = Tc
Kd = Ki/4 如果使用微分项

一旦确定了初始增益，将它们转化为整数。过程增益 K 等于输入的 PV 计算值变化量除以输出的 CV 计算值变化量，不是 PV 或 CV 工程值变化量。将时间以秒为单位计算。一旦 Kp, Ki 和 Kd 确定了，可以将 Kp 和 Kd 乘以 100，以整型数方式输入，Ki 则乘以 1000，输入到用户参数变量队列中。

例子

下面的 PID 实例采样时间为 100 毫秒，Kp 增益为 4.00，Ki 增益为 1.500。设定值存储在%R0001，控制变量输出存储在%AQ0002，过程变量来自%AI0003。CV 上下限幅必须设定，这个例子中设为 20000 和 4000，可选的小死区为+5 和 -5。40 个字的变量数列从%R0100 开始。用户参数一般在变量数列中设定。但是设定%M0006 即可用逻辑中存储的常数重新初始化从%R0102(%Ref+2)开始的 14 个字。(一个有用的技巧)

可以用%M1 将 PID 块转换为手动模式，这种模式下可以调整手动命令字%R113 的值。%M4 或者%M5 可以用于增加或减少%R113 和 PID 以及积分器的值，每 100 毫秒增/减 1。对于快速的手动操作，%M2 / %M3 接通，则 CPU 在每个扫描周期内，用%R113 内的值加/减 %R2 内的值，最后再将结果赋给%R113。PID OK 时，%T1 输出 ON。


1
V_M00006
BLK CLR WORD
35
V_R00100
IN
BLKMOV INT
10 IN1 Q V_R00102
5 IN2
5 IN3
400 IN4
0 IN5
1500 IN6
0 IN7
BLKMOV INT
2000 IN1 Q V_R00109
400 IN2
0 IN3
0 IN4
0 IN5
0 IN6
0 IN7
V_T00010
2
PID IND
V_R001...
V_R00001 SP CV V_R00600
V_R00503 PV
V_M00001 MA
V_M00004 UP
V_M00005 DN
V_T00001
3
V_M00002
ADD INT
V_R00113 IN1 Q V_R00113
V_R00002 IN2
4
V_M00003
SUB INT
V_R00113 IN1 Q V_R00113
V_R00002 IN2

Chapter 11 结构化文本

结构化文本(ST)程序语言是一种 IEC 61131-3 文本化语言：对那些有高级语言(例如 C 语言)编程经验的人来说非常方便。用结构化文本语言编程，有更大的弹性，而且很容易在不同类型的控制器之间转换。他的简洁性是你能够在一个屏幕上看到复杂的运算法则。

本章描述结构化文本如何在 PAC 系统上执行。关于如何在编程软件上使用结构化文本编程器，参考在线帮助。

模块类型的程序块，参数化程序块和函数块可以在 ST 中编写。_MAIN 程序块也可以在 ST 中编写。关于块的细节，参考第六章"程序组织"

语言总览

语句

结构化文本由一系列的语句组成，这些语句又由表达式和语言关键字构成。一个语句会令 PLC 执行某个特别的动作。语句包含变量分配，条件评估，反复和调用内部函数的能力。PAC 系统支持 11-4 页“语句类型”中所描述的语句

表达式

表达式计算变量和常数的值。一个表达式可能包括逻辑运算，变量和常数。一个简单表达式的例子如***(x + 5)。***

复杂表达式可以由简单表达式嵌套而成，例如：

(a + b) * (c + d) – 3.0 ** 4.

逻辑运算

下表列出了可以在表达式中使用的逻辑运算。根据逻辑运算优先级列出，优先级决定了在一个表达式内的执行次序。优先级最高的逻辑运算首先应用，之后是优先级稍低的。优先级相同的，按从左至右的顺序执行。同一组的逻辑运算，如+和-，有相同的优先级

LD 中使用的地址逻辑运算可以用作 ST 操作数。地址逻辑运算包括非直接地址(例如，@Var1)，数组(例如，Var1[3])，字地址中的位(例如, Var1.X[3])，和结构域(例如, Var1.field1)

优先级	*逻辑运算 r*	*操作数类型*	*描述*
Group 1 (Highest)	(…)		括号
Group 2	- NOT	INT, DINT, REAL BOOL, BYTE, WORD, DWORD	取反 二进制非
Group 3	**,^	INT, DINT, UINT, REAL[1]	求幂[3,5]
Group 4	* / MOD	INT, DINT, UINT, REAL INT, DINT, UINT, REAL INT, DINT, UINT	乘[3] 除[2,3] 模[2]
Group 5	+ -	INT, DINT, UINT, REAL INT, UINT, DINT, REAL	加[3] 减[3]
Group 6	<, >, <=, >=	INT, DINT, UINT, REAL, BYTE, WORD, DWORD	比较
Group 7	= <>, !=	ANY[4] ANY[4]	等于 不等
Group 8	AND, &	BOOL, BYTE, WORD, DWORD	二进制与
Group 9	XOR	BOOL, BYTE, WORD, DWORD	二进制或
Group 10 (Lowest)	OR	BOOL, BYTE, WORD, DWORD	二进制非

[1] 使用**操作数的表达式的计算结果为 REAL 型。基数必须为 REAL 型。幂次可以为 INT, DINT, UINT 或 REAL 型

[2] 将 0 除错误标记为应用程序错误。

[3] 使用数学操作数可能导致上/下溢出。上溢出部分会被删除。

[4] 操作数类型标为 ANY 的逻辑运算可以使用支持的任何类型的操作数。支持的操作数类型包括：BOOL, INT, DINT, UINT, BYTE, WORD, DWORD 和 REAL。不支持使用 STRING 和 TIME 类型操作数。

[5] 如果操作数为正无穷或负无穷，则结果不确定。

操作数类型

不支持类型重塑。要转换类型，必须使用内置的转换函数。内置转换函数在 11-6 页“函数调用”中描述。

对于无类型的逻辑运算，操作数的类型必须匹配。

结构化文本的语法

ST 在 PAC 系统中的语法遵从 IEC 61131-3 标准：

- 结构化文本语句必须以半角的分号(;)结束。
- 结构化文本的变量必须在项目的变量表中声明。

这些符号有以下功能：

:=将一个表达式赋给一个变量

;表明语句结束

[ ]数组索引为整数时，指明元素位置。例如，将 j+10 赋给数组第三个元素：
intarray[3]: = j + 10;

(* *) 表示注释。注释可以跨行。例如

**(*This comment spans
multiple lines.*)**

//或者 **‘** 则表示单行注释。例如：

c :=a+b; *//*这是一个单行注释

c :=a+b; ‘这是一个单行注释

语句类型

表明实际程序执行的结构化文本语句有以下类型：

语句类型	*描述*	*例子*
赋值语句	给对象赋一个值	A := 1; B := A; C := A + B;
函数调用	调用并执行函数	FbInst(IN1 := 1, OUT1 => A);
返回	使程序从子程序中返回。返回语句能使程序更早的从子程序块中退出。	RETURN;
退出	在满足最终条件 (条件为 TRUE (1) 时)前终止循环	EXIT;
IF	表示一个或多个语句有条件的执行	IF (A < B) THEN C := 4; ELSIF (A = B) THEN C:= 5; ELSE C := 6 END_IF;
WHILE	在条件不满足 (条件=FALSE (0))前，重复执行一个语句序列。	WHILE J <= 100 DO J := J + 2; END_WHILE;
重复	重复执行一个语句序列，直到二进制表达式为真 (TRUE (1))	REPEAT J := J + 2; UNTIL J >= 100 END_REPEAT;
空语句		;

赋值语句

赋值语句将表达式计算结果赋给变量(数据类型要相同)

注意:

- 赋值语句可以影响转换位
- 赋值语句将覆盖位计入在内

格式

```
Variable := Expression;
```

在这里:
Variable 可以是一个简单的变量，也可以是数组的元素等等
Expression 是单个数值，表达式或者复杂表达式

例子

二进制赋值语句：

```
VarBool1 := 1;
VarBool2 := (val <= 75);
```

数组元素赋值：

```
Array_1[13] := (RealA /RealB)* PI;
```

函数调用

结构化文本函数调用执行一个预先定义的预算，这个运算完成数学，位串或者其他操作。函数调用由函数或程序块名称，以及需要的输入输出参数组成。

结构化文本逻辑可以调用程序块和下表所示的 PAC 系统内置的函数。函数调用必须是单独的一个语句，不能是嵌套表达式的一部分。关于 PAC 系统内置函数，参考第八章"指令设定参考"。

调用某些函数，如通讯请求(COMM_REQ)，需要使用命令块或者参数块。对于这些函数，首先声明数组，在逻辑内进行初始化，然后作为一个参数传给函数。

可供 ST 调用使用的内置函数

Note: 当前的 PAC 系统版本只支持下表所列举的函数。不支持其他内值函数

高级数学函数	ASIN, ATAN, ACOS, COS, SIN, TAN, DEG_TO_RAD, RAD_TO_DEG LOG, LN, EXP, EXPT, SQRT_INT, SQRT_DINT, SQRT_REAL ABS_INT, ABS_DINT, ABS_REAL SCALE_DINT, SCALE_INT, SCALE_UINT
控制	SWITCH_POS, SUS_IO, DO_IO, SVC_REQ,
数据转换	BCD4_TO_INT, BCD4_TO_UINT, BCD4_TO_REAL BCD8_TO_DINT, BCD8_TO_REAL DINT_TO_BCD8, DINT_TO_INT, DINT_TO_UINT, DINT_TO_REAL UINT_TO_BCD4, UINT_TO_BCD8, UINT_TO_INT, UINT_TO_DINT INT_TO_BCD4, INT_TO_BCD8, INT_TO_UINT, INT_TO_DINT REAL_TO_INT, REAL_TO_UINT, REAL_TO_DINT TRUNC_INT, TRUNC_DINT
数据移动	COMM_REQ,

可供 ST 调用的程序块类型

可供 ST 调用 的块包括类型块，参数化块或函数块

正规调用/非正规调用

PAC 系统支持的正规和非正规 ST 调用

正常调用	*非正规调用*
输入参数赋值使用':='符号 输出参数赋值使用'=>'符号	输入输出参数在括号内列出
可选参数可以省略	不能省略参数
参数可以是任何顺序	参数顺序必须正确，正确顺序如下： 输入 临时位置 (需要的话) 参数长度(需要的话) 输出，从最后一个输出参数开始.
ENO 在正规函数或程序块调用中标明 内置函数或者用户自定义程序块可选择使用 ENO 输出参数标志函数或程序块的连续性。	在非正规函数或程序块调用中，没有特别标明 ENO 参数

正规函数调用的格式

```
FunctionName(IN1 := inparam1, IN2 := inparam2, OUT1 => outparam1, ENO =>
enoparam);
```

非正规函数调用的格式

```
FunctionName(inparam1, inparam2, outparam1);
```

例子

下面这段代码显示了 TAN 函数调用：

```
TAN( AnyReal, Result );
```

返回语句

返回语句使程序能早一点从子程序块中退出。例如，在下面几行代码中，第三行不执行。变量 a 的最终值为 4。

```
a := 4;
RETURN;
a := 5;
```

IF 语句

IF 结构是有条件的执行语句。条件由二进制表达式确定，条件满足则执行一个语句序列。IF 结构包含两个可选部分，ELSE 和 ELSIF，这两个可选部分使你能够执行其他的语句序列。每个 IF 结构可以有一个 ELSE 语句和几个 ELSIF 语句。

格式

```
IF  BooleanExpression1  THEN
    StatementList1;
[ELSIF  BooleanExpression2  THEN  (*Optional*)
    StatementList2;]
[ELSE                                     (*Optional*)
    StatementList3;]
END_IF;
```

这里:

BooleanExpression 任何可以得到二进制值的表达式

StatementList 任何结构化文本语句

注意: ELSIF 或者 ELSEIF 都可以作为 IF 语句的 else if 子句

操作

如果两个可选部分都使用，则有下列的判断：

- 如果 BooleanExpression1 为 TRUE (1)，执行 StatementList1。之后执行关键字 END_IF 后的语句
- 如果 BooleanExpression1 为 FALSE (0)，并且 BooleanExpression2 为 TRUE (1)，执行 StatementList2。之后执行关键字 END_IF 后的语句
- 如果两个二进制表达式(BooleanExpression)都为 FALSE (0)，则执行 StatementList3。之后执行关键字 END_IF 后的语句

如果一个可选项没出现，则程序继续执行关键字 END_IF 后的语句。

例子

下面这段代码将文本写入变量状态，依赖于 I/O 输入点的输入值

```
IF Input01 < 10.0 THEN
    Status := Low_Limit_Warning;
ELSIF Input02 > 90.0 THEN
    Status := Upper_Limit_Warning;
ELSE
    Status := Limits_OK;
END_IF;
```

WHILE 语句

只要设定的条件满足 (TRUE (1)), WHILE 语句就重复执行 WHILE…END_WHILE 中间的语句。WHILE 语句首先检查条件是否满足，之后有条件的执行语句序列。当语句序列不是必须执行时，这种循环结构非常有用。

格式

```
WHILE  <BooleanExpression>  DO
    <StatementList>;
END_WHILE;
```

这里:

BooleanExpression	任何可以得到二进制值的表达式
StatementList	任何结构化文本语句

操作

如果 BooleanExpression 为 FALSE (0)，立刻退出循环；否则，如果 BooleanExpression 为 TRUE (1)，则重复执行 StatementList。由于开始时就检查二进制表达式的值，所以可能永远都不执行 StatementList。

注意: 如果出现死循环，则会造成看门狗定时器超时。要避免出现死循环。

例子

下面这段代码在 J 小于 100 时每次将 J 的值加 2.

```
WHILE J <= 100 DO
    J := J + 2;
END_WHILE;
```

重复(REPEAT)语句

重复语句在退出条件满足之前反复执行 REPEAT…END_REPEAT 之间的语句。他首先执行语句序列，然后检查退出条件。这种循环结构适用于语句序列至少要执行一次的情况。

格式

```
REPEAT
    StatementList;
UNTIL  BooleanExpression  END_REPEAT;
```

这里:

BooleanExpression	任何可以得到二进制值的表达式
StatementList	任何结构化文本语句

操作

执行 StatementList。如果 BooleanExpression 为 FALSE (0)，重复执行循环；否则，如果 BooleanExpression 为 TRUE (1)，退出循环。由于循环的最后才检查二进制表达式的值，语句序列最少执行一次。

注意: 如果出现死循环，则会造成看门狗定时器超时。要避免出现死循环。

例子

下面这段代码从数组读取数值，直到 Value > 5 为止(或者到达数组上限)。因为最少要读取一个数组值，所以要使用 REPEAT 循环。本李子中所有的变量类型为 DINT, UINT 或者 INT.

```
Index :=1;

REPEAT
    Value:= @Index;
    Index:=Index+1;
UNTIL  Value > 5  OR Index >= UpperBound END_REPEAT;
```

退出语句

退出语句用于提前终止循环(WHILE, REPEAT)的执行。程序继续执行循环终结语句(END_WHILE, END_REPEAT)后面的语句。退出语句的典型使用是用在 IF 语句中。

格式

```
EXIT;
```

这里:

用退出条件表达式来确定是否提前退出。

例子

下面这段代码显示了如何使用退出语句。当变量值等于 10 的时候，退出 WHILE 循环并且继续执行 END_WHILE 后面的语句。

```
while (1) do
  a := a + 1;
  IF (a = 10) THEN
     EXIT;
  END_IF;
END_WHILE;
```

Chapter 12 通讯

本章描述了 PAC 系统 CPU 的以太网和串口通讯特性。其中包含以下信息：

以太网通讯

关于 PAC 系统以太网通讯的详细信息，请参考以下手册：

PAC 系统 TCP/IP 以太网通讯用户指南, GFK-2224

PAC 系统 TCP/IP 通讯站管理器手册, GFK-2225

内置以太网接口

RX7i CPU 有一个内置的以太网接口，CPU 可以通过这个以太网口与其他控制系统或编程软件进行 TCP/IP 通讯。这些通讯是使用独有的 SRTP 协议在 TCP/IP(Internet)的第四层上进行的。以太网接口也支持通过 UDP(用户自寻址协议)的以太网全局数据协议。

内置以太网接口有两个 RJ-45 接口。这两个接口可以单独或者同时接到其他以太网接口设备上。每个口自动感知数据传输速率(10Mbps 或 100Mbps)，双工模式(全双工或者半双工)以及电缆连接方式(直连还是交叉连接)。

注意

两个以太网口不能直接或间接连到同一个设备上。以太网上的 HUB 或转换器连接必须形成树状，否则会产生重复的数据包。

10Base-T/100Base-Tx 口针脚分布

针脚号	*信号*	*描述*
1	TD+	发出数据 +
2	TD-	发出数据 -
3	RD+	接收数据 +
4	NC	没连接
5	NC	没连接
6	RD-	接收数据-
7	NC	没连接
8	NC	没连接

以太网接口模块

RX7i 和 RX3i 支持基于机架的以太网接口模块。(这些模块不可互换) 关于以太网接口模块的容量，安装和操作信息，参考 PAC 系统的 *TCP/IP 以太网通讯* GFK-2224 和 PAC 系统站管理器 GFK-2225。

类型	*目录号*	*描述*
RX7i	IC698ETM001	以太网 VME 模块
RX3i	IC695ETM001	以太网 PCI 模块

串行通讯

CPU 板上的独立串口可以通过前面板的连接器来访问。端口 1 和端口 2 位外部设备提供了串行接口。端口 1 用于固件更新。RX7i CPU 提供了用作以太网站管理器接口的第三个串口。所有的串口是相互隔离的。

串口通讯能力

端口 1 和端口 2 被配置为下列几种模式中的一种。关于 CPU 配置的详细信息，参考第三章。

- RTU Slave 端口用于 Modbus RTU slave 协议。这种模式也允许 SNP 主控器连接端口，SNP 主控器包括 Winloader 用户或者编程软件。详见第十三章“串行 I/O, RTU 和 SNP 协议”
- 端口可以被用户逻辑访问。这使 C 块能够通过 C Runtime 库函数进行串口 I/O 操作。
- Available –端口不被 CPU 固件使用
- SNP Slave –端口只能使用 SNP slave 协议。详见第十三章“串行 I/O,RTU 和 SNP 协议”
- Serial I/O –通过使用 COMMREQ 函数，在这个端口上进行一般意义上的串行通讯。详见第十三章“串行 I/O,RTU 和 SNP 协议”

支持的特性

特性	*端口 1 (Com 1)*	*端口 2 (Com 2)*	*端口 3 (站管理器) 只适用于 RX7i*
RTU Slave 协议	Yes	Yes	No
SNP Slave	Yes	Yes	No
Serial I/O – 与 COMMREQ 同时使用	Yes	Yes	No
固件更新 (Winloader 用户)	CPU 在停止/无 IO 模式	No	No
Message 模式 –只用于 C 块 (C Runtime 库函数: 串行读写, sscanf, sprintf)	Yes	Yes	No
站管理器(只用于 RX7i)	No	No	Yes
RS-232	Yes	No	Yes
RS-485	No	Yes	No

可配置的停止模式协议

你可以基于以上已配置的端口(运行模式)协议配置用于停止模式的协议。运行/停止协议转换要单独为每个串口配置。

运行模式协议确定了那些协议可选作停止模式协议。如果没有选择停止模式协议，则使用缺省的停止模式协议。详见第三章“端口 1 和端口 2 参数”

串口针脚分配

端口 1

端口 1 是光电隔离的 RS-232 兼容口。他有一个标准针脚的 9 针，D-sub 的母接头。这是一个 DCE（数据通讯设备）口，允许通过直连电缆和标准 AT-类型的 RS-232 端口连接。

端口 1 RS-232 信号

针脚号	信号名称	描述
1*	NC	没有连接
2	TXD	发送数据
3	RXD	接收数据
4	DSR	数据设定准备好
5	0V	信号地
6	DTR	数据终端准备好
7	CTS	清除发送
8	RTS	要求发送
9	NC	没有连接

* 从模块的前面看去，针脚 1 在连接器的底端右侧。

Port 2

端口 2 是光电隔离的 RS-485 兼容口。他有一个 15 针，D-sub 的母接头。这个端口不支持 RS-485 转 RS-232 适配器(IC690ACC901)。这是一个 DCE 口。

端口 2 RS-485 信号– RX7i CPUs

这个端口不提供+5V 电压，因此 RS-485 到 RS-232 的转换需要自带电源的转换器。

针脚号.	*信号名称*	*描述*
1*	Shield	电缆屏蔽
2	NC	没有连接
3	NC	没有连接
4	NC	没有连接
5	NC	没有连接
6	RTS(A)	差分传送请求
7	0V	信号地
8	CTS(B‘)	差分传送清除
9	RT	电阻终结
10	RD(A‘)	差分接收数据
11	RD(B‘)	差分接收数据
12	SD(A)	差分发出数据
13	SD(B)	差分发出数据
14	RTS(B’)	差分传送请求
15	CTS(A’)	差分传送清除

*　从模块的前面看去，针脚 1 在连接器的底端右侧。

端口 2 RS-485 信号– RX3i CPUs

针脚号	*信号名称*	*描述*
1*	Shield	电缆屏蔽
2	NC	没有连接
3	NC	没有连接
4	NC	没有连接
5	+5VDC	逻辑电源**
6	RTS(A)	差分传送请求
7	0V	信号地
8	CTS(B‘)	差分传送清除
9	RT	电阻终结
10	RD(A‘)	差分接收数据
11	RD(B‘)	差分接收数据
12	SD(A)	差分发出数据
13	SD(B)	差分发出数据
14	RTS(B’)	差分传送请求
15	CTS(A’)	差分传送清除

*从模块的前面看去，针脚 1 在连接器的底端右侧。

**针脚 5 为外部选项提供隔离的+5VDC 电源(最大 300mA).

端口 3 (只适用于 RX7i)

端口 3，站管理器口，是一个隔离的 RS-232F 兼容口。他有一个 9 针，D 型的母接头。这是一个 DCE（数据通讯设备）口，允许通过直连电缆和标准 AT-类型的 RS-232 端口连接。为了将来使用点对点协议(PPP)，这个端口使用了标准 RS-232 的全部信号。

站管理器 RS-232 信号

针脚号	信号名称	描述
1*	DCD	数据载波检测
2	TXD	发送数据
3	RXD	接收数据
4	DSR	数据设定准备好
5	0V	信号地
6	DTR	数据终结准备好
7	CTS	清除发送
8	RTS	请求发送
9	RI	环形指示器

*从模块的前面看去，针脚 1 在连接器的底端右侧。

串行电缆长度和屏蔽

CPU 的串口到计算机或其他设备的串口通过串行电缆连接。这种连接可以用 IC690ACC901 电缆工具包来做，也为了特别应用而做。

最大电缆长度(CPU 到最后一个串行连接设备之间的串行电缆长度)为：

端口 1 (RS-232) = 15 米 (50 英尺) –可选用屏蔽电缆

端口 2 (RS-485) = 1200 米(4000 英尺) –可选用屏蔽电缆

端口 3 (RS-232) = 15 米(50 英尺) –可选用屏蔽电缆

串行接口波特率

协议	***端口 1 (RS-232)***	***端口 2 (RS-485)***	***站管理器 (端口 3) (RS-232)***
RTU 协议	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	不支持
通过 WinLoader 进行固件更新	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	Not supported	不支持
Message 模式	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	不支持
SNP Slave	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	不支持
Serial I/O	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K	不支持

90-70 系列通讯与可选择的智能模块

PAC 系统 RX7i 支持下列的 90-70 系列通讯和智能可选择模块

- 通讯协处理模块(CMM), IC697CMM711
- 可编程协处理模块(PCM), IC697PCM711
- DLAN/DLAN+ (驱动本地网络)接口, IC697BEM763

通讯协处理模块(CMM)

1.50 和更高版本的 PAC 系统 RX7i CPU 支持固件版本为 4.20 或更高的 IC697CMM711 模块。你必须确定你使用的是一个正确的固件版本的 CMM，因为 CPU 不能检测 CMM 的固件版本。(EEPROM 上附带的标签上没有模块的固件版本号)

PAC 系统不支持在 IC697CMM711 上使用以下特性：

- 访问符号变量
- WAIT 模式 COMMREQs.
- 通过 CMM 的串口连接 CPU 和编程软件
- 永久自寻址数据表

下列限制只在 IC697CMM711 和 PAC 系统一起使用时有效：

- 部分支持访问%W 变量。只有偏移地址在 0—65535 范围内的%W 可以通过 CMM 访问。
- 对 PAC 系统来说当前程序名永远是 LDPROG1
- 读写的变量超出了变量表限定的范围，将会报一个次要错误，错误号 90(REF_OUT_OF_RANGE)，而不是像 90-70 一样报 F4 (INVALID_PARAMETER) 故障
- 在 ERROR NACK 情况下，将控制程序号，权力等级和其他的后背状态数据设为 0
- 与 CMM 通讯时，PAC 系统 CPU 将 90-70 CPX935 主要/次要类型(主要类型 12, 次要类型 35)传给 CMM 的可修改存储器
- PAC 系统中控制程序号返回值为 01，90-70 系列控制程序号返回值为 FF
- 如果 RX7i 应用程序需要访问 CMM 的双端口寄存器，可以使用 BUS READ 和 BUS WRITE 函数。访问 CMM 时，将函数块的 Region 参数设为 1。(对于 CMM 来说, region 1 预先被设定为模块的全部双端口寄存器)

注意: 关于 IC697CMM711 操作的细节，参考 PAC 系统串行通讯和 90 系列用户手册，GFK-0582

可编程协处理模块(PCM)

1.50 和更高版本的 PAC 系统 RX7i CPU 支持固件版本为 4.05 或更高的 IC697PCM711 模块。你必须确定你使用的是一个正确的固件版本的 PCM，因为 CPU 不能检测 PCM 的固件版本。(EEPROM 上附带的标签上没有模块的固件版本号)

PAC 系统不支持在 IC697PCM711 上使用以下特性：

- 通过 PCM 的串口连接 CPU 和编程软件
- 访问符号变量
- WAIT 模式 COMMREQs.
- 不支持下列的 C 函数：
  - chk_genius_bus
  - chk_genius_device
  - get_cpu_type_rev
  - get_memtype_sizes
  - get_one_rackfault
  - get_rack_slot_faults
- C 函数 write_dev 不会向只读变量(%S 变量,转换位和覆盖位)写入。如果做这种尝试，运行时调用会失败并且返回错误代码。

下列限制只在 IC697PCM711 和 PAC 系统一起使用时有效：

- 部分支持访问%W 变量。只有偏移地址在 0—65535 范围内的%W 可以通过 PCM 访问。
- 对 PAC 系统来说当前程序名永远是 LDPROG
- 在 ERROR NACK 情况下，将控制程序号，权力等级和其他的后背状态数据设为 0.
- 如果 PCM 上的应用程序访问 VME 总线，程序使用的 VME 地址必须符合 PAC 系统 RX7i VME 地址分配规则。PAC 系统 RX7i VME 地址分配在 PAC 系统 RX7i 用户指南的 VME 模块集成部分讲述，GFK-2235
- 与 PCM 通讯时，PAC 系统 CPU 将 90-70 CPX935 主要/次要类型(主要类型 12, 次要类型 35)传给 PCM 的可修改存储器
- 如果 RX7i 应用程序需要访问 PCM 的双端口寄存器，可以使用 BUS READ 和 BUS WRITE 函数。访问 PCM 时，将函数块的 Region 参数设为 1。(对于 PCM 来说, region 1 预先被设定为模块的全部双端口寄存器)

注意: 关于 IC697PCM711 操作的细节，参考可编程协处理器模块和支持软件，GFK-0255.

DLAN/DLAN+ (驱动本地网络)接口

1.50 和更高版本的 PAC 系统 RX7i CPU 支持固件版本为 3.00 或更高的 IC697BEM763 模块。你必须确定你使用的是一个正确的固件版本的 PCM，因为 CPU 不能检测 DLAN 的固件版本。(EEPROM 上附带的标签上没有模块的固件版本号)

如果 RX7i 应用程序需要访问 DLAN 的双端口寄存器，可以使用 BUS READ 和 BUS WRITE 函数。访问 DLAN 模块时，将函数块的 Region 参数设为 1。(对于 DLAN 来说, region 1 预先被设定为模块的全部双端口寄存器)

注意: DLAN 接口模块是使用范围有限的专用模块。如果你有一个 DLAN 系统，详细情况请参考 *DLAN/DLAN+接口模块用户手册，*GFK-0729

13 章 *串行I/O，SNP 和RTU 协议*

本章描述的是串行 I/O 特征，该特征可以用来直接通过应用程序控制 CPU 串口 1 和 2 的读/写行为。

本章也包含用 COMM_REQ 来配置用于 SNP、RTU 或串行 I/O 协议的 CPU 串行口指令。

- 用 COMM_REQ 功能配置串行口
  - 带有一个附加程序器的 RTU 从/SNP 从 操作
  - 配置 SNP 协议的 COMM_REQ 命令块
  - 配置 RTU 协议的 COMM_REQ 数据块
  - 配置串行 I/O 的 COMM_REQ 数据块
- 串行 I/O COMM_REQ 命令
  - 初始化端口
  - 建立输入缓冲器
  - 清除输入缓冲器
  - 读端口状态
  - 写端口控制
  - 取消操作
  - 自动拨号
  - 写字节
  - 读字节
  - 读字符串
- RTU 从协议
- SNP 从协议

RTU 和 SNP 协议的详细描述见 *Serial Communications User's Manual* (GFK-0582)。

使用 COMM_REQ 功能配置串行口

串行 I/O 通过使用 COMM_REQ 功能实现。协议的操作，如通过串口发送字符或等待一个输入字符等，是通过 COMM_REQ 功能块实现。

COMM_REQ 要求所有命令数据在执行之前以一个正确的顺序（在一个命令块里）放在 CPU 存储器中。COMM_REQ 应该通过一个单触发线圈的触点来执行，防止多次发送数据。用于 COMM_REQ 功能的操作数和命令块格式的详细资料，查阅第 8 章“指令系统说明”。

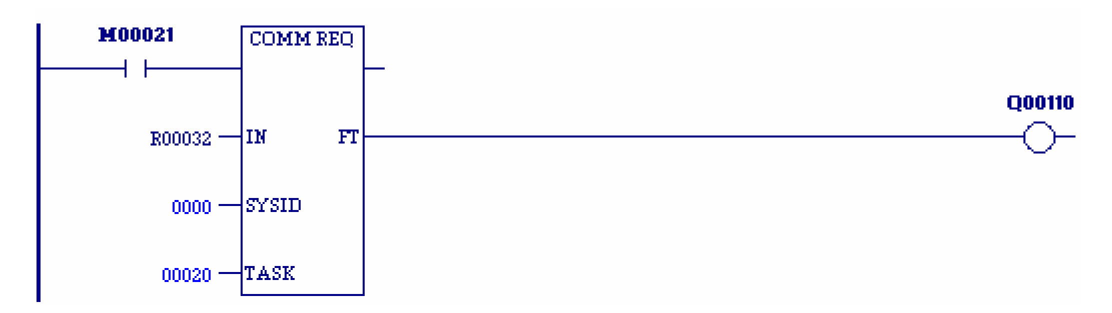
COMM_REQ 用下列任务来指定端口执行想要的操作：

task 19 for port 1
task 20 for port 2

注意: 因为地址偏移量存储在一个 16 位的字组，全部的%W 存储器类型不能用于 COMM_REQ。

COMM_REQ 功能举例

该例中，当%M0021 是 ON，一个开始于%R0032 的命令块被送到 CPU（0 机箱，0 槽）端口 2（通讯任务 20）。如果运行 COMM_REQ 时有错误发生，%Q0110 被置位。



时序

如果一个端口配置 COMM_REQ 被发送到一个串口，通常该串口有一个 SNP 主设备（例如，程序器）与之连接， COMM_REQ 功能回送一个错误代码到 COMM_REQ 状态字。

发送另一个 COMM_REQ 到相同的端口

发送一个 COMM_REQ 来配置一个串口后，应用程序应该监控 COMM_REQ 状态字，以便确定什么时间可以开始发送协议专用 COMM_REQ 到该端口。 推荐优先应用清除 COMM_REQ 状态字，而不是改变配置。当请求处理完成，状态字将被置一个非 0 值。

无效的端口配置组合

对于 RX3i CPU, 与硬件配置下载相反，ME 程序软件安全装置阻止程序器与 CPU 之间频繁通讯。这是因为 RX3i 以太网模块可能不存在或被删除，在这种情况下就需要一个用于程序器通讯的串行连接。对于 RX7i CPU，以太网端口在 CPU 模块上，所以以太网总是对程序器通讯可用的。

COMM_REQ 命令块参数值

下表列出了在用于配置串口的 COMM_REQ 命令块中使用的公共参数值。所有的数都是十进制。

参数	值
协议选择器	1 = SNP 3 = RTU 5 = Serial I/O 7 = 信息模式
数据速率	0 = 300 1 = 600 2 = 1200 3 = 2400 4 = 4800 5 = 9600 6 = 19200 7 = 38400 8 = 57600 9 = 115200
奇偶	0 = None 1 = 奇数 2 =偶数
流控制	0 = 硬件[RTS / CTS] 1 = None 2 = 软件[XON / XOFF] (只用于 Serial I/O)
每字符位数	0 = 7 位 1 = 8 位
停止位数	0 = 1 个停止位 1 = 2 个停止位
双向传输模式	0 = 2 线 1 = 4 线 2 = 4 线传送器总是在 ON 状态。
翻转延时 (只用于 SNP)（这个翻转表示数据传输方向的改变）	0 = None 1 = 10 ms 2 = 100 ms 3 = 500 ms
暂停(只用于 SNP)	0 =长 (8 sec) 1 =中等 (2 sec) 2 =短 (500 ms) 3 = “None” (200 ms)

抽样 COMM_REQ 命令块

下面的 COMM_REQ 命令块提供配置不同的协议的例子。除后面有 H 的表示十六进制外，所有的值都是十进制的。

注意：例子不由信息模式提供，但和 Serial I/O 例子相似，可以用一个命令块来设置，用于协议选择器时值是 7。

配置 SNP 协议实例命令块

	值	含义
Address	16	数据块长度
Address + 1	0 =没有等待(不支持 WAIT 模式)	WAIT/NOWAIT 标志
Address + 2	0008 = %R, 寄存器存储器	状态字指针存储器类型
Address + 3	给出 COMM_REQ 状态字地址数，从 0 开始。（例如，状态字地址是 100，给出的值是 99）	状态字指针偏移量
Address + 4	不用	等待暂停值
Address + 5	不用	最长通讯时间
Address + 6	FFF0H	命令字 (串口设置)
Address + 7	1 = SNP	协议
Address + 8	0 =从	端口模式
Address + 9	见 13-3 页“COMM_REQ 命令块参数值”	数据速率
Address + 10	0 = None, 1 = 奇, 2 =偶	奇偶
Address + 11	不用 (SNP 缺省值总只选择 NONE)	流控制
Address + 12	0 = None, 1 = 10ms, 2 = 100ms, 3 = 500ms	翻转延时
Address + 13	0 = 长, 1 = 中等, 2 = 短, 3 = None	暂停
Address + 14	不用 (SNP 缺省总是选择 8 位)	每字符位数
Address + 15	0 = 1 个停止位, 1 = 2 个停止位	停止位数
Address + 16	不用	接口
Address + 17	不用 (SNP 缺省总是选择 4 线模式)	双向模式
Address + 18	用户提供*	设备标识符 字节 1 和 2
Address + 19	用户提供*	设备标识符 字节 3 和 4
Address + 20	用户提供*	设备标识符 字节 5 和 6
Address + 21	用户提供*	设备标识符 字节 7 和 8

* SNP 从端口的设备标识符用字的最低有效字节的最低有效字符充填进字里。例如，如果前两个字符是“A”和“B”，Address + 18 将由 16 进制数 4241 构成。

配置 RTU 协议 COMM_REQ 实例数据块

	值	*含义*
Address	13	数据块长度
Address + 1	0 =没有等待(不支持 WAIT 模式)	WAIT/NOWAIT 标志
Address + 2	0008 = %R, 寄存器存储器	状态字指针存储器类型
Address + 3	给出 COMM_REQ 状态字地址数，从 0 开始。（例如，状态字地址是 100，给出的值是 99）	状态字指针 偏移量
Address + 4	不用	等待暂停值
Address + 5	不用	最长通讯时间
Address + 6	FFF0H	命令字 (串口设置)
Address + 7	3 = RTU	协议
Address + 8	0 = 从	端口模式
Address + 9	见 13-3 页“COMM_REQ 命令块参数值”	数据速率
Address + 10	0 = None, 1 = 奇, 2 =偶	奇偶
Address + 11	0 =硬件, 1 = None	流控制
Address + 12	不用	翻转延时
Address + 13	不用	暂停
Address + 14	不用 (RTU 缺省总是选择 8 位)	每字符位数
Address + 15	不用(RTU 缺省总是选择一个停止位)	停止位数
Address + 16	不用	接口
Address + 17	0 = 2 线, 1 = 4 线, 2 = 4 线传送器总是在 ON 状态	双向模式
Address + 18	局地址(1-247)	设备标识符

配置 Serial I/O 协议 COMM_REQ 实例数据块

	Values	**Meaning**
Address	12	数据块长度
Address + 1	0 =没有等待(不支持 WAIT 模式)	WAIT/NOWAIT 标志
Address + 2	0008 = %R, 寄存器存储器	状态字指针存储器类型
Address + 3	给出 COMM_REQ 状态字地址数，从 0 开始。（例如，状态字地址是 100，给出的值是 99）	状态字指针偏移量
Address + 4	不用	等待暂停值
Address + 5	不用	最长通讯时间
Address + 6	FFF0H	命令字 (串口设置)
Address + 7	5 = Serial I/O	协议
Address + 8	不用	端口模式
Address + 9	见 13-3 页“COMM_REQ 命令块参数值”	数据速率
Address + 10	0 = None, 1 = Odd, 2 = Even	奇偶
Address + 11	0 = 硬件, 1 = None, 2 = 软件	流控制
Address + 12	不用	翻转延时
Address + 13	不用	暂停
Address + 14	0=7 位, 1=8 位	每字符位数
Address + 15	0 = 1 个停止位, 1 = 2 个停止位	停止位数
Address + 16	不用	接口
Address + 17	0 = 2 线, 1 = 4 线, 2 = 4 线传送器总是在 ON 状态	双向模式

从 CPU 扫描中调用 Serial I/O COMM_REQ

用 Serial I/O COMM_REQ 执行串行协议可以被扫描时间限制。例如，如果协议要求从远程设备请求一个确定信息时，远程设备必须在接收到请求信息 5ms 内开始响应，当扫描时间是 5ms 或更长时这个方法可能不会成功，因为不能保证及时响应。

Serial I/O 协议只有在 CPU 运行时才能激活，因为它完全是被应用程序中 COMM_REQ 功能驱动。当 CPU 停止时，串行 I/O 的端口配置将恢复到停止模式协议。按照 CPU 配置的端口设置中指定的停止模式，停止模式可以由 RTU 从 或 SNP 从来设置；缺省是 RTU 从。

兼容性

Serial I/O 支持的 COMM_REQ 功能块不被当前现有的协议支持（像 SNP 从和 RTU 从等）。如果这些 COMM_REQ 功能块被用于一个由不支持 COMM_REQ 功能块的协议配置的端口，送回错误信息。

Serial I/O COMM_REQ 的状态字

COMM_REQ 成功完成，一个 1 送回 COMM_REQ 状态字。其他任何送回的值都是错误代码，低字节是主要错误代码，高字节是次要错误代码。

主要错误代码	*描述*	
1 (01h)	**成功比较** (在状态字里这是期望完成值。).	
12 (0Ch)	**本地错误**—处理本地命令的错误。次要错误代码识别具体错误。	
	2 (02h)	COMM_REQ 命令不被支持
	6 (06h)	无效的 PLC 存储器指定类型.
	7 (07h)	无效的 PLC 存储器偏指定移量
	8 (08h)	不能访问 PLC 存储器
	12 (0Ch)	COMM_REQ 数据块长度太。小
	14 (0Eh)	COMM_REQ 数据无效
	15 (0Fh)	不能分配系统资源来完成 COMM_REQ.
13 (0Dh)	**远程错误** —处理远程命令的错误。次要错误代码识别错误。	
	2 (02h)	被请求读的字节数大于输入缓冲器范围、被请求写的字节数为 0 或大于 250 个字节
	3 (03h)	COMM_REQ 数据块长度太小.字符串数据没有或不完整。
	4 (04h)	等待连续接收数据时接收暂停
	6 (06h)	无效的 PLC 存储器指定类型.
	7 (07h)	无效的 PLC 存储器偏指定移量
	8 (08h)	不能访问 PLC 存储器
	12 (0Ch)	COMM_REQ 数据块长度太小
	16 (10h)	操作系统服务错误。用来执行请求的操作系统已经送回一个错误
	17 (11h)	端口设备错误. 用来执行服务的设备已经检测一个错误。一个中断被接收或一个 URAT 错误（奇偶、成帧、超限）存在。
	18 (12h)	取消请求. 请求在完成之前被取消。
	48 (30h)	字符串行输出暂停。串行口不能传输字符串。（当串口被配置用于硬件流控制时，可能是没有 CTS 信号）
14 (0Eh)	**自动拨号错误** —在企图发送一个命令字符串给附加外部调制解调器时有一个错误存在。次要错误代码识别具体错误。	
	2 (02h)	调制解调器命令字符串长度超过参考存储器类型的极限。
	3 (03h)	COMM_REQ 数据块长度太小，输出命令字符串数据丢失或不完整。
	4 (04h)	串行输出暂停。串口不能传输调制解调器自动拨号输出。
	5 (05h)	没有来自调制解调器的响应。检查调制解调器和电缆。
	6 (06h)	调制解调器响应“BUSY”。调制解调器不能完成请求的连接。远程调制解调器在使用中，稍后再试连接请求。
	7 (07h)	调制解调器响应“NO CARRIER” 调制解调器不能 完成请求的连接。检查本地和远程调制解调器及电话线。
	8 (08h)	调制解调器响应“NO DIALTONE”. 调制解调器不能 完成请求的连接。检查调制解调器接线和电话线
	9 (09h)	调制解调器响应“ ERROR”.调制解调器不能 完成请求的命令。检查调制解调器命令和调制解调器。
	10 (0Ah)	调制解调器“RING”,表示调制解调器正被另一个调制解调器呼叫。调制解调器不能 完成请求的命令。稍后再试调制解调器命令。
	11 (0Bh)	调制解调器不知名的响应。调制解调器不能完成请求。检查调制解调器命令串和调制解调器。响应应该是 CONNECT 或 OK。

Serial I/O COMM_REQ 命令

下列的 COMM_REQ 用来执行 Serial I/O:

- 本地 COMM_REQ – 不通过串口接收或发送数据。
  - 初始化端口(4300)
  - 建立输入缓冲器 (4301)
  - 清除输入缓冲器 (4302)
  - 读端口状态 (4303)
  - 写端口控制(4304)
  - 取消操作 (4399)
- 远程 COMM_REQ - 通过串口接收或发送数据。
  - 自动拨号(4400)
  - 写字节 (4401)
  - 读字节(4402)
  - 读字符串(4403)

重叠 COMM_REQ

一些 Serial I/O COMM_REQ 在可以处理另一个 COMM_REQ 之前必须执行完毕。而另一些则可以不这样。

必须完全执行的 COMM_REQ

- 自动拨号(4400)
- 初始化端口(4300)
- 建立输入缓冲器(4301)
- 清除输入缓冲器(4302)
- 读端口状态 (4303)
- 写端口控制(4304)
- 取消操作 (4399)
- 串口设置 (FFF0)

在另外的 COMM_REQ 执行时可以是不完全执行的 COMM_REQ

下表显示的是当其他 COMM_REQ 执行时，写字节、读字节和读字符串 COMM_REQ 是否可以没有完全执行。

当前未完全执行的 COMM_REQ	新执行的 COMM_REQ										
	自动拨号 (4400)	写字节 (4401)	初始化端口(4300)	建立输入缓冲器 (4301)	清除输入缓冲器(4302)	读端口状态(4303)	写端口控制(4304	读字节 (4402)	读字符串 (4403)	取消操作 n (4399)	串口设置 (FFF0)
写字节 (4401)	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
读字节(4402)	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes	No
读字符串 (4403)	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes	No

初始化端口功能 (4300)

这个功能导致一个复位命令被发送到指定端口。同时取消当前正在进行 COMM_REQ，清除内部输入缓冲器。RTS 设置为停止。

初始化端口功能实例命令块

	值 (十进制)	值 (十六进制)	含义
address	0001	0001	数据块长度
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4300	10CC	初始化端口命令

操作注意事项

由于该命令执行而取消调用的远程 COMM_REQ 将送回一个 COMM_REQ 状态字表示请求取消。 (次要代码 12H).

警告

如果在一个写字节（**4401**）**COMM_REQ** 正在从一个串口传输一个字符串时，该 **COMM_REQ** 被发送，传输停止。字符串里传输停止的位置是不确定的。另外，**CPU** 正在发往的设备接收的最后字符也是不确定的。

建立输入缓冲器功能 (4301)

这个功能为传统应用程序提供兼容性。在 PACSystems Serial I/O 的执行中，内部输入缓冲器总是设为 2K 字节。该功能给 COMM_REQ 状态字送回一个成功状态，不管在命令块中缓冲器被指定的长度。

数据一从串口被接收就被放进输入缓冲器。如果输入缓冲器满，任何从串口被接收的额外数据将被丢弃，端口状态字（见“读端口状态功能”）“溢出错误”位被置位。

从缓冲器中恢复数据

数据可以通过“读字符串”或“读字节”功能从缓冲器中被恢复。数据不能从应用程序中直接得到。

如果数据不能从缓冲器中及时恢复，可能有一些字符丢失。

建立输入缓冲器功能实例命令块

	值 (十进制)	值 (十六进制)	含义
address	0002	0002	数据块长度
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4301	10CD	设置输入缓冲器命令
address +7	0064	0040	缓冲器长度(以字为单位)

清除输入状态字功能(4302)

这个操作清空输入缓冲器内通过串口接收的但还没有用读命令检索的任何字符。所有这样的字符都被丢失。

清除输入缓冲器功能的实例命令块

	值 (十进制)	值 (十六进制)	含义
address	0001	0001	数据块长度
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4302	10CE	清除输入缓冲器命令

读端口状态功能(4303)

该功能送回当前端口状态。下列事件能被检测到：

1. 起先有一个读请求，被请求的字符数现在被收到或者指定的暂停时间已过。
2. 起先有一个写请求，指定字符数的传输已完成或暂停时间已过。

功能送回的状态表示事件已完成。如果起先读、写请求都有，不止一种情形会同时发生。

读端口状态功能的实例命令块

	值 (十进制)	值 (十六进制)	含义
address	0003	0003	数据块长度
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4303	10CF	读端口状态命令
address +7	0076	004C	端口状态存储器类型 (%M)
address +8	0101	0065	端口状态存储器偏移量 (%M101)

端口状态

端口状态由一个状态字和输入缓冲器里还没有被应用程序检索的字符数组成（字符已被接收且是有效的）。

word 1	端口状态字(见下面)
word 2	缓冲器里字符有效

端口状态字可以是:

位	*名称*	*定义*	*含义*	
15	RI	读正在处理	Set	“读字节”或“读字符串” 被调用
			Cleared	早先的“读字节”或“读字符串已经暂停或被取消或完成
14	RS	读成功	Set	“读字节”或“读字符串已经成功完成
			Cleared	新的“读字节”或“读字符串被调用
13	RT	读暂停	Set	在“读字节”或“读字符串期间收到暂停发生信息
			Cleared	新的“读字节”或“读字符串被调用
12	WI	写正在处理	Set	新的“写字节”被调用
			Cleared	先前被调用的“写字节” 已经超时或被取消或完成
11	WS	写成功	Set	先前被调用的“写字节” 已经成功完成
			Cleared	新的“写字节”被调用
10	WT	写超时	Set	在写字节期间传输超时发生
			Cleared	新的“写字节”被调用
9	CA	字符有效	Set	未读字符在缓冲器中
			Cleared	没有未读字符在缓冲器中
8	OF	溢出错误	Set	在串口或内部缓冲器中有溢出错误发生
			Cleared	“读端口状态“被调用
7	FE	帧错误	Set	在串口上有帧错误发生
			Cleared	“读端口状态“被调用
6	PE	奇偶错误	Set	在串口上有奇偶错误发生
			Cleared	“读端口状态“被调用
5	CT	CTS 激活	Set	在串口上 CTS 线激活或端口没有 CTS 线
			Cleared	在串口上 CTS 线不激活
4 - 0	U	不用, 应该是 0		

写端口控制功能(4304)

这个功能为指定端口强制 RTS：

写端口控制功能实例命令块

	值 (十进制)	值 (十六进制)	含义
address	0002	0002	数据块长度
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4304	10D0	写端口控制命令
address +7	xxxx	xxxx	端口控制字

端口控制字

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTS	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

端口控制字可以是:

15	RTS	RTS 输出的被命令状态 1 = 激活 RTS 0 = RTS 激活解除
0-14	U	不用 (应该为 0)

操作注意事项:

对于 CPU 端口 2 (RS-485), RTS 信号也受发送驱动器控制。因此，RTS 的控制由发送驱动器的当前状态决定。如果发送驱动器不激活，在串行线上，带有写端口控制 COMM_REQ 的坚持 RTS 将使 RTS 不被坚持。发送驱动器的状态受协议控制，由当前端口的双向模式决定。对于 2 线或 4 线双向模式，发送驱动器只有在发送期间被激活。因此，当正在发送数据时，在串行线上的 RTS 只有在端口 2（配置为 2 线或 4 线双向模式）上才看到激活。对于点对点双向模式，发送驱动器总是激活的。因此，在点对点双向模式里，在串行线上的 RTS 将总是反映用写端口控制 COMM_REQ 选择了什么。

取消 COMM_REQ 功能 (4399)

该功能取消当前正在处理的操作。该功能可以用来取消读操作和写操作。

如果读操作正在处理，有未被处理的字符在输入缓冲器里，这些字符被留在缓冲器中，以后读时也是有效的。串口不被复位。

取消操作功能实例命令块

	值 (十进制)	*值 (十六进制)*	*含义*
address	0002	0002	数据块长度
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4399	112F	取消操作命令
address +7	0001	0001	取消的处理类型 1 所有操作 2 读操作 3 写操作

操作注意事项:

由于该命令执行而调用的远程 COMM_REQ 将送回一个 COMM_REQ 状态字表示请求取消（次要代码 12H）。

警告

当一个写字节(4401) COMM_REQ 正在从一个串口发送一个字符串，如果 “Cancel ”COMM_REQ 被送进 Cancel All 或 Cancel Write 模式, 字符串发送停止。字符串里的位置（字符串发送停止位置）是不确定的。另外，接收 CPU 正在发送字符的设备接收的最后一个字符也是不确定的。t

自动拨号功能 (4400)

这个功能允许 CPU 自动拨一个调制解调器，发送一个指定字节串。

为了实现这个作用，端口必须被配置成可用于 Serial I/O.。自动拨号功能执行且和调制解调器连接成功之后，可以使用其他 Serial I/O 功能 (Write Bytes, Set Up Input Buffer, Flush Input Buffer, Read Port Status, Write Port Control, Read Bytes, Read String, 和 Cancel Operation)。

例子

使用 3 个命令可以完成寻呼机发音，需要 3 个 COMM_REQ 命令块:

自动拨号: 04400 (1130h)	拨调制解调器.
写字节: 04401 (1131h)	指定一个 1 到 250 个字节长的 ASCII 码串从串口发出
自动拨号: 04400 (1130h)	PLC 应用程序负责挂断电话连接，通过重新发出自动拨号命令和发送挂断命令字符串完成。

自动拨号命令块

自动拨号命令自动发送一个遵守贺氏协定的换码顺序。如果正在使用的调制解调器不支持贺氏协定，可以用写字节命令来拨调制解调器。

一般用于贺氏兼容调制解调器的命令字符串的例子列一下表：

命令串	长度	功能
ATDP15035559999<CR>	16 (10h)	脉冲拨打号码 1-503-555-9999
ATDT15035559999<CR>	16 (10h)	音频 拨打号码 1-503-555-9999
ATDT9,15035559999<CR>	18 (12h)	音频 拨号，以暂停方式使用外线
ATH0<CR>	5 (05h)	挂断电话
ATZ <CR>	4 (04h)	恢复调制解调器配置到内部存储值

抽样自动拨号命令块

这个 COMM_REQ 命令块使用 贺氏兼容调制解调器拨打号码 234-5678。.

字	定义	值
1	0009h	CUSTOM 数据块长度(包括命令字符串)
2	0000h	NOWAIT 模式
3	0008h	状态字存储器类型 (%R)
4	0000h	状态字地址减 1 (%R0001)
5	0000h	不用

6	0000h	不用	
7	04400 (1130h)	自动拨号命令数	
8	00030 (001Eh)	调制解调器响应超时(30s)	
9	0012 (000Ch)	命令字符串的字节数	
10	5441h	A (41h), T (54h)	
11	5444h	D (44h), T (54h)	
12	3332h	电话号码	2 (32h), 3 (33h)
13	3534h		4 (34h), 5 (35h)
14	3736h		6 (36h), 7 (37h)
15	0D38h		8 (38h) <CR> (0Dh)

写字节功能 (4401)

这个功能可以用来通过指定串口向远程设备发送一个或更多的字符。被发送的字符必须在字基准存储器里。操作完成之前字符不应该被改变。

每次执行这个操作可以发送至多 250 个字符。在字符没有全部发送之前或没有超时发生，操作一直进行（例如，如果硬件流控制一直被使用，远程设备从不激活发送功能）。

写字节功能的实例命令块

	值 (十进制)	值 (十六进制)	含义
address	0006	0006	数据块长度（包括发送的字符）
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4401	1131	写字节命令
address +7	0030	001E	传输超时 (30s). 看下面的注释.
address +8	0005	0005	要写的字节数
address +9	25960	6568	‘h’ (68h), ‘e’ (65h)
address +10	27756	6C6C	‘l’ (6Ch), ‘l’ (6Ch)
address +11	0111	006F	‘o’ (6Fh)

尽管在这个例中使用可打印的 ASCII 字符，但对可以发送的字符的值没有限制。

操作注意事项;

指定 0 作为发送超时设置实际发送数据需要的总时间的值，加上 4 秒。

警告

在写字节 **COMM_REQ** 正在从一个串口传输一个字符串时，如果在 **Cancel All** 或 **Cancel Write** 模式下，发送初始化端口（**4300**）**COMM_REQ** 或取消操作（**4399**）**COMM_REQ**，发送停止。字符串里的位置（在这个位置字符串传输停止）是不确定的。另外，接收 **CPU** 正在发送字符的设备接收的最后一个字符也是不确定的。

读字节功能(4402)

这个功能 使一个或一个以上的字符从指定端口被读出。这些字符从内部缓冲器被读出，放在指定的内部数据存储区域。该功能送回检索到的字符数和仍然在输入缓冲器中未经处理的字符数。如果请求 输入字符数为 0，那么只有输入缓冲器中未经处理的字符数被送回。

如果没有足够的字符可用来确保读字节请求且指定被读的字符数为一个非零值，那么在足够的字符被接收完成或暂停时间到之前，操作将不能完成。读字节操作完成时，端口状态显示读操作完成的原因（字符被接收完成或暂停时间到）。状态字在读操作完成（暂停或所有的数据已经被接收）之前不被刷新。

如果暂停时间被置为 0， 读字节 COMM_REQ 保持在待处理状态直到被请求的数据接收完或该功能被取消。

如果这个 COMM_REQ 由于某种原因失败，没有数据被送回到输入数据区。从内部输入缓冲器读出的数据保留，可以被后面的读请求检索到。

读字节功能的实例命令块

	值 (十进制)	*值 (十六进制)*	*含义*
address	0005	0005	数据块长度（包括发送的字符）
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4402	1132	读字节命令
address +7	0030	001E	读暂停（30s）
address +8	0005	0005	读的字节数
address +9	0008	0008	输入数据存储器类型 (%R).
address +10	0100	0064	输入数据存储器地址(%R0100)

读字节功能的返回数据处格式

返回数据由实际读的字符数、读完成后输入缓冲器中仍然有用的字符数（如果有的话）和实际输入字符组成。

Address	实际读的字符数
Address + 1	输入缓冲器中仍然有用的字符数（如果有的话）
Address + 2	头两个字符 (第一个字符在低字节)
Address + 3	第三和第四个字符(第三字符在低字节)
Address + n	后面的字符

读字节操作注意事项：

如果输入数据存储器类型参数被指定为一个字存储器类型，如果实际接收的字节数是奇数，那么用接收的数据写进的最后一个字的高字节无变化地被留下。I

从串口被接收的数据被放在内部输入缓冲器中。如果缓冲器满，从串口接收的任何数据都被丢弃，端口状态字（见读端口状态功能）的溢出错误位被置位。

读字符串功能 (4403)

这个功能使字符从指定端口被读，直到指定的结束字符被接收。把从内部输入缓冲器读到字符放入指定输入数据区域。

该功能送回检索到的字符数和输入缓冲器中仍未处理的字符数。如果请求 输入字符数为 0，那么只有输入缓冲器中未经处理的字符数被送回。

如果结束字符不在输入缓冲器中，那么在结束字符已经被接收或暂停时间到之前，操作将不能完成。读字符串操作完成时，端口状态显示读操作完成的原因（结束字符已经被接收或暂停时间到）。

如果暂停时间被置为 0， 读字符串 COMM_REQ 保持在待处理状态直到被请求的字符串接收完或由指定结束字符终止。

如果这个 COMM_REQ 由于某种原因失败，没有数据被送回到输入数据区。从内部输入缓冲器读出的数据保留，可以被后面的读请求检索。

读字符串功能的实例命令块

	值 (十进制)	*值 (十六进制)*	*含义*
address	0005	0005	数据块长度（包括发送的字符）
address +1	0000	0000	NOWAIT 模式
address +2	0008	0008	状态字存储器类型 (%R)
address +3	0000	0000	状态字地址减 1 (%R0001)
address +4	0000	0000	不用
address +5	0000	0000	不用
address +6	4403	1133	读字符串命令
address +7	0030	001E	读暂停（30s）
address +8	0013	000D	结束字符 (回车): 必须在 0 和 255 (0xFF)之间, 包括 0 和 255
address +9	0008	0008	输入数据存储器类型 (%R).
address +10	0100	0064	输入数据存储器地址(%R0100)

读字符串功能的返回数据处格式

返回数据由实际读的字符数、读完成后输入缓冲器中仍然有用的字符数（如果有的话）和实际输入字符组成。

Address	实际读的字符数
Address + 1	输入缓冲器中仍然有用的字符数，（如果有的话）
Address + 2	头两个字符 (第一个字符在低字节)
Address + 3	第三和第四个字符(第三字符在低字节)
Address + n	后面的字符

读字符串操作注意事项

如果输入数据存储器类型参数被指定为一个字存储器类型，如果实际接收的字节数是奇数，那么用接收的数据写进的最后一个字的高字节无变化地被留下。I

从串口被接收的数据被放在内部输入缓冲器中。如果缓冲器满，从串口接收的任何数据都被丢弃，端口状态字（见读端口状态功能）的溢出错误位被置位

RTU 从协议

RTU 协议是一个用于 RTU 设备和主机之间通讯查询响应协议。 主机能用 RTU 协议通讯。主机是主设备，它发送一个查询到一个 RTU 从设备，从设备响应主设备。RTU 从设备不能查询，只能响应主设备。一个 PACSystems CPU 只能作一个 RTU 从设备使用。

RTU 转移的数据由带有一位可选奇偶位的 8 位二进制字符组成。没有控制字符加到数据块中；每个查询和响应的最后字段包含一个错误校验（循环冗余校验）以确保数据的正确传输。

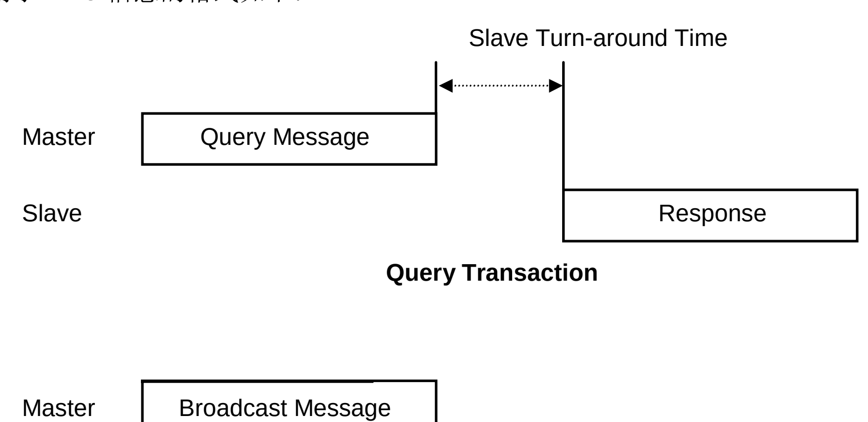
注意: 应该避免把局地址 1 用于一个 PACSystems 控制系统中任何其他的 Modbus 从设备，因为 PACSystems CPU 缺省的局地址就是 1。CPU 在两种情况下使用缺省地址：

1. 如果没有配置时上电，使用缺省局地址 1。
2. 当端口模式参数被设置为信息模式，Modbus 变成停止模式下的协议，局地址默认为 1，在 CPU 配置中给端口指定一个停止模式除外，然后改变用于停止模式局地址。

在以上任何一种情况下，如果有一个局地址 1 配置从设备，当 PACSystems 响应从设备的请求时容易引起混乱。

信息格式

一般用于 RTU 信息的格式如下：



RTU 信息格式

主设备通过发送一个查询或广播请求信息开始发送数据。如果主设备发送一个查询信息到一个从设备，从设备通过发送一个响应信息来表示完成数据传输完成。当主设备发送的是广播请求时，则从设备没有响应信息发送。

RTU 从处理时间

在查询结束到响应查询开始的这一段时间叫做从设备处理时间。一个 PACSystem 从设备的设备处理时间取决于控制器通讯窗口时间和扫描时间。RTU 请求只有在控制器通讯窗口里被处理。在正常扫描模式下，每一次扫描控制器通讯窗口出现一次。因为在 PACSystem 中，扫描时间最多可达 2.5s，所以处理一个 RTU 请求的时间最多是 2.5s。另一个因素是硬件配置允许的控制器通讯窗口时间。如果配置的是一个非常小的控制器通讯窗口，RTU 请求可能一次扫描完成不了，导致 RTU 处理需要多个扫描周期。详情查阅第 5 章“CPU 窗口模式”。

信息类型

RTU 协议有四中信息类型：查询、正常响应、错误响应和广播。

查询

主设备发送一个查询信息到一个从设备。

正常响应

在从设备执行查询所请求的功能之后，发回一个针对该功能的正常响应，表示请求成功。

错误响应

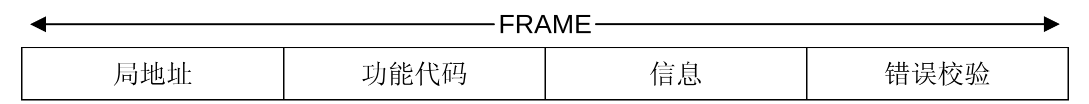
从设备接收到查询，但不能执行所请求的功能。从设备送回一个错误响应，指出请求不能执行的原因。（有些错误没有错误信息送出。更多信息见“通讯错误”。）

广播

主设备通过地址 0 向所有从设备发送一个信息。所有接收广播信息的从设备执行相应的请求功能。这条指令通过主设备内的一个暂停结束。

信息字段

典型信息的信息字段如下图所示。下一节做具体说明。



局地址

局地址是被选择的用于数据传输的从站地址。长一个字节，值是 0 到 247（包括 0 和 247）。地址 0 选择所有的从站，表示这是一个广播信息。从 1 到 247 就代表选择的从站局地址。

功能代码

功能代码确定发给从站的命令。一个字节长，由 0 到 255 来定义，如下表所示：

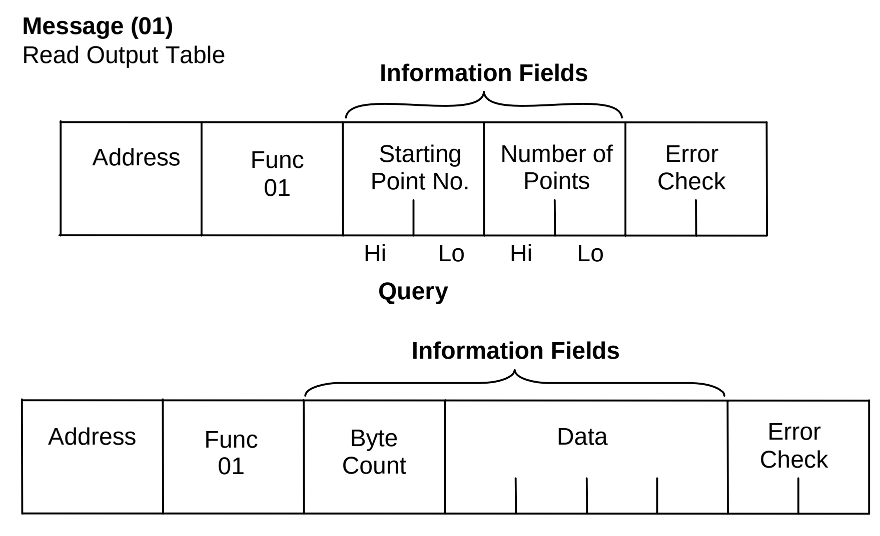
功能代码	*描述*
0	非法功能
1	读输出表
2	读输入
3	读寄存器
4	读模拟量输入
5	强制单个输出
6	预置单个寄存器
7	读例外状态
8	回送维护
9-14	不支持功能
15	强制多点
16	预置多个寄存器
17	报告设备类型
18–21	不支持功能
22	屏蔽写 4x 寄存器
23	读/写 4x 寄存器
24–66	不支持功能
67	读暂时存储器
68-127	不支持功能
128-255	为例外的响应保留

信息字段

所有与局地址字段、功能代码字段和错误校验字段不同的字段，一般被叫做“信息”字段。信息字段包含要求指定的或回答被请求功能的附加信息。不同类型的信息有不同类型或数量的信息字段。（详细资料见 13-32 页，在“信息描述”中关于每个信息类型和功能代码的信息字段查找。）有些信息(Message 07 Query 和 Message 17 Query)没有信息字段。

例子

见下图，信息 *READ OUTPUT TABLE (01) Query* 的信息字段由开始点编号字段和点数字段组成。信息 *READ OUTPUT TABLE (01) Response* 的信息字段由字节计数字段和数据字段组成。



Normal Response

信息字段例子

一些信息字段包括在 RTU 从设备内被访问数据范围的条目。

注意: 数据地址从 0 开始。这意味着在 RTU 信息指定数据地址时必须用实际地址减 1。对于上例中的信息（01）*READ OUTPUT TABLE Query*. ，在开始点编号字段指定开始地址。为了指定%Q0001 作为开始地址，应该把地址%Q0000 放在这个字段里。放在点数字段里值决定有多少从地址%Q0001 开始的 Q 位被读。例如：

- 开始点编号字段 = %Q0007, 所以开始地址是 %Q0008
- 点数字段 = 16 (0010h), 所以地址%Q0008 到%Q0023 将被读

错误校验字段

错误校验字段有两字节长，包含一个循环冗余校验(CRC-16)代码。它的值 是一个局地址内容的功能、功能代码和信息字段。产生 CRC-16 代码的详细资料在 13-28 页“循环冗余校验”中描述。注意信息字段长度是一个变量。为了正常产生 CRC-16 代码，帧长必须确定。关于计算每种定义过的功能代码的帧长，见 13-31 页“计算帧长”。

信息长度

信息长度随着信息类型和被发送的数据的总量的改变而改变。确定一个信息信息长度的有关内容在“信息描述”中查找。

字符格式

一个信息以一个字符串的形式发送。信息中的每个字节作为一个字符发送。下图显示的就是字符格式。一个字符由开始位（0）、8 个数据位、一个可选的奇偶位和一个停止位组成。两个字符之间的连线保持 1 状态。

		MSB			Data Bits				LSB	
10	9	8	7	6	5	4	3	2	1	0
Stop	Parity (optional)									Start

信息终止

每个站监控字符之间的时间。当一个三个字符时间的周期过去了，仍然没有接收到字符，那么信息结束是假的。采用下一个接收的字符作为一个新信息的开始。当首先发生下面两个事件之一，帧结束出现：

- 接收的字符数等于计算的帧的长度。
- 一个四个字符时间的长度过去了，没有接收到字符。

暂停用法

暂停用于错误检查和错误恢复的串行连接，并防止信息的结尾和信息顺序丢失。注意尽管模块允许在接收的信息中的每个字符之间多达 3 个字符传输时间，但在模块传输的信息中的字符之间只有不超过一个字符时间的一半。主设备发送一个查询信息之后，在从设备没有回答这个请求之前，主设备应该等待一个从设备的处理时间。从设备处理时间受控制器窗口时间和 CPU 扫描时间的影响。具体描述在 13-24 页“RTU 从设备处理时间”。

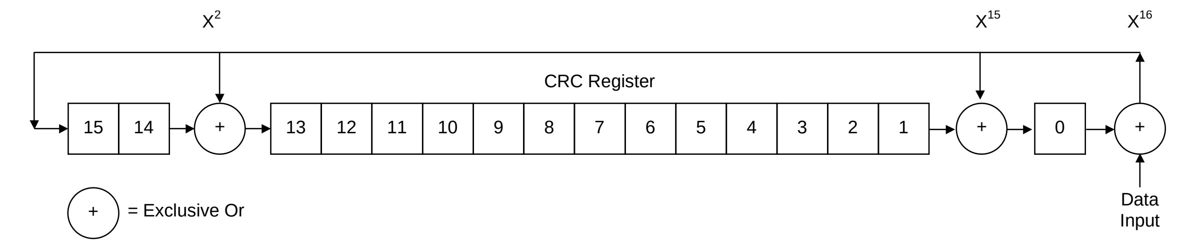
循环冗余校验(CRC)

CRC 是校验错误最有效的系统之一。CRC 由产生于发送装置的两个校验字符组成，这两个字符加在被发送数据结尾。用相同的方法，接收装置对于接受的数据产生自身的 CRC，并把它和发送器产生的 CRC 进行比较，确保数据正确发送。在这一节里不做 CRC 的完整的数学推导。有关这方面的内容可在很多关于数据通讯的文章中查找。在计算 CRC 中应该理解的基本步骤如下：

- CRC 里的位数乘上组成信息的数据位数。
- 结果除以生成的多项式（使用无进位求 2 的摸数方法），余数就是 CRC。
- 忽视商，把余数（CRC）加进数据位，把该信息和 CRC 一起发送。
- 接收装置用该信息加上 CRC 去除以生成的多项式，如果余数是 0，发送就是无错误发送。

一个生成多项式用 X 幂的代数和表示，如 $X^3 + X^2 + X^0$(或 1)，这个多项式转换成二进制数就是 1101。只要发送装置和接收装置使用相同的值，一个生成多项式可以是任意长，包含任何 1 或 0 的形式。几个已经展开的标准生成多项式最适合错误检测。RTU 协议用多项式 $X^{16} + X^{15} + X^2 + 1$，用二进制表示就是 1 1000 0000 0000 0101。这个多项式生成的 CRC 就是众所周知的 CRC-16。

上面的讨论可通过硬件或软件实现。硬件实现需要建立一个多段的基于生成多项式的移位寄存器。



循环冗余校验寄存器

为了生成 CRC，信息数据位被送进移位寄存器，一次一位。寄存器包含一个预设值。当每个数据位都已经移进移位寄存器时，所有的位都向右移。LSB 和数据位进行 XOR 运算，结果是：和位 1 原来的内容 XOR（结果放在位 0），和位 14 原来的内容 XOR（结果放在位 13），最后 LSB 被移进位 15。这样重复执行，直到一个信息里的所有数据位都被处理完。CRC-16 软件执行将在下一节中叙述。

计算 CRC-16

下面给出了计算 CRC-16 的伪码：

被发送数据的预设字节数

初始化 16 位余数（CRC）寄存器全部为 1

把第一个 8 位数据字节和 16 位 CRC 寄存器的高位字节进行 XOR 运算。结果就是当前 CRC。

INIT SHIFT: 初始化移位计数器为 0.

SHIFT 当前 CRC 寄存器向右移位一位。.

移位计数器加 1。

移出右边的位（标志）是 1 还是 0？

如果是 1，把生成多项式和当前的 CRC 做 XOR 运算

如果是 0，继续。

移位计数器等于 8 吗?

NO, 返回 SHIFT.

YES,字节数加 1

字节数大于数据长度吗？

NO,把下一个 8 位数据字节和当前 CRC 做 XOR 运算，进入 INIT SHIFT.

YES, 当前 CRC 加到数据信息的尾部发送，然后退出。

当信息被发送，接收装置对所有的数据位和发送 CRC 执行相同的 CRC 操作，。如果信息准确被接收，余数（CRC ）为 0。

CRC-16 计算举例

RTU 设备首先发送最右边的字节（寄存器的或离散数据的）。发送 CRC-16 的第一位是 MSB。因此，例中 CRC 多项式的 MSB 在最右边。X^{16}项被丢掉，因为，它只影响商（被丢弃）而不影响余数（CRC 字符）。生成的多项式是 1010 0000 0000 0001。把余数全部初始化为 1。

例中, 计算 CRC-16 用于 RTU 信息,读例外状态 07。信息格式如下：

地址	功能	CRC-16
01	07	

例中，设备 1（地址是 01）被查询。必须要知道发送数据的总量，相关的资料到“计算帧长度”中查找。本例中数据长度是 2 个字节。

发送装置 CRC-16 运算法则

	MSB[2]		*LSB*[2]		*Flag*
初始化余数	1111	1111	1111	1111	
XOR 第一个数据字节	0000	0000	0000	0001	
当前 CRC	1111	1111	1111	1111	
Shift 1	0111	1111	1111	1111	0
Shift 2	0011	1111	1111	1111	1
XOR 生成多项式	1010	0000	0000	0001	
当前 CRC	1001	1111	1111	1110	
Shift 3	0100	1111	1111	1111	0
Shift 4	0010	0111	1111	1111	1
XOR 生成多项式	1010	0000	0000	0001	
当前 CRC	1000	0111	1111	1110	
Shift 5	0100	0011	1111	1111	0
Shift 6	0010	0001	1111	1111	1
XOR 生成多项式	1010	0000	0000	0001	
当前 CRC	1000	0001	1111	1110	
Shift 7	0100	0000	1111	1111	0
Shift 8	0010	0000	0111	1111	1
XOR 生成多项式	1010	0000	0000	0001	
当前 CRC	1000	0000	0111	1110	
XOR 2nd data byte	0000	0000	0000	0111	
当前 CRC	1000	0000	0111	1001	
Shift 1	0100	0000	0011	1100	1
XOR 生成多项式	1010	0000	0000	0001	
当前 CRC	1110	0000	0011	1101	
Shift 2	0111	0000	0001	1110	1
XOR 生成多项式	1010	0000	0000	0001	
Current CRC	1101	0000	0001	1111	
Shift 3	0110	1000	0000	1111	1
XOR 生成多项式	1010	0000	0000	0001	
Current CRC	1100	1000	0000	1110	
Shift 4	0110	0100	0000	0111	0
Shift 5	0011	0010	0000	0011	1
XORv	1010	0000	0000	0001	
当前 CRC	1001	0010	0000	0010	
Shift 6	0100	1001	0000	0001	0
Shift 7	0010	0100	1000	0000	1
XOR 生成多项式	1010	0000	0000	0001	
当前 CRC	1000	0100	1000	0001	
Shift 8	0100	0010	0100	0000	1
XOR 生成多项式	1010	0000	0000	0001	
发送 CRC	1110	0010	0100	0001	
	E	2	4	1	

接收装置 1 CRC-16 运算法则

	MSB[2]		*LSB*[2]		*Flag*
在数据后接收 CRC	1110	0010	0100	0001	
XOR CRC 第一个字节	0000	0000	0100	0001	
当前 CRC	1110	0010	0000	0000	
Shift 1	0111	0001	0000	0000	0
Shift 2	0011	1000	1000	0000	0
Shift 3	0001	1100	0100	0000	0
Shift 4	0000	1110	0010	0000	0
Shift 5	0000	0111	0001	0000	0
Shift 6	0000	0011	1000	1000	0
Shift 7	0000	0001	1100	0100	0
Shift 8	0000	0000	1110	0010	0
XOR CRC 第二个字节	0000	0000	1110	0010	
当前 CRC	0000	0000	0000	0000	
Shift 1-8 yields	0000	0000	0000	0000	

All errors for receiver final CRC-16 indicates transmission correct.

1. 接收装置和发送装置一样通过相同的 CRC 运算法则处理接收到的数据。例中，接收在所有数据位后开始，但发送 CRC 没有被正确接收。因此，在这个点，接收 CRC 应该等于发送 CRC。如果是这样，CRC 运算输出将是 0，表示传输正确。

   和 CRC 一起发送的信息将是：

地址	*功能*	***CRC–16***	
01	07	41	E2

2. MSB 和 LSB 参考只是数据字节，不是 CRC 字节。CRC MSB 和 LSB 位与数据位的顺序相反。

计算帧长

为了产生任何信息的 CRC-16，必须知道信息的长度。任何类型的长度可以根据下面的数据表中确定。

RTU 信息长度

功能代码和名称		*查询或广播信息长度（小于 **CRC** 代码）*	*响应信息长度（小于 **CRC** 代码）*
0		未定义	未定义
1	读输出表	6	3 + 3rd byte*
2	读输入表	6	3 + 3rd byte*
3	读寄存器	6	3 + 3rd byte*
4	读模拟量输入	6	3 + 3rd byte*
5	强制单个输出	6	6
6	预设单个寄存器	6	6
7	读例外状态	2	3
8	回送/维护	6	6
9-14		未定义	未定义
15	强制多个输出	7 + 7th byte*	6
16	预设多个寄存器	7 + 7th byte*	6
17	报告设备类型	2	8
18-21		未定义	未定义
22	Mask Write 4x Registers	8	8
23	Read/Write 4x Registers	13+byte 11*	5+byte 3*
24–66		未定义	未定义
67	Read Scratch Pad	6	3 + 3rd byte*
68-127		未定义	未定义
128-255		未定义	3

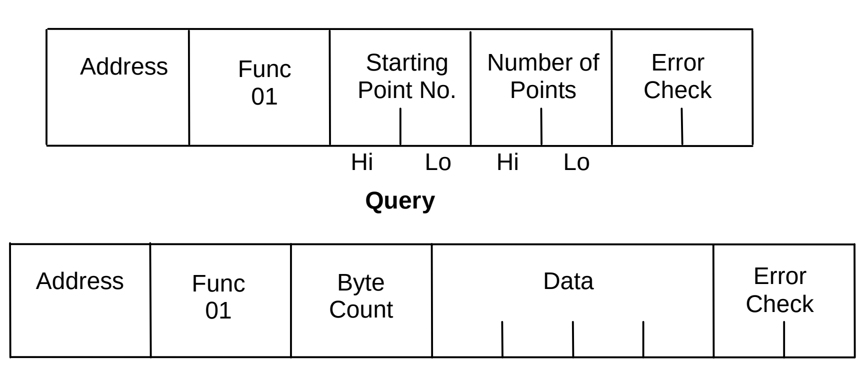
*这个字节的值是被发送数据里的字节数。

RTU 信息描述

本节介绍每个 RTU 信息的格式和字段。

信息 (01): 读输出表

格式:



查询:

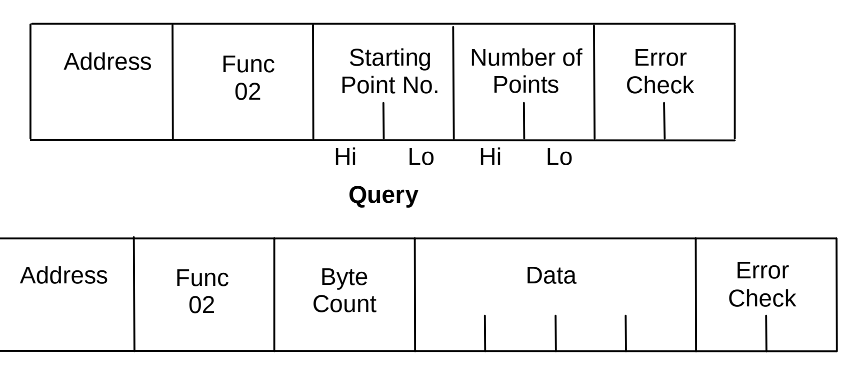
- 不允许使用地址 0，因为这不能是一个广播请求。
- 功能代码是 01.
- 开始点编号是双字节长，可以是任何小于附属 CPU 里最高输出点有效编号的值。开始点编号等于一个比第一个输出点编号小 1 的数，这个输出点是对该请求的正常响应送回的。
- 点数值是双字节长，指定一个正常响应送回的输出点数。开始点值及点数值的总和必须小于或等于附属 CPU 里最高输出点数变量的值。开始点编号的高位字节和字节字段数作为第一个字节被发送。低位字节是第二个字节。

响应:

- 字节计数是一个从 1 到 256（0=256）的二进制数。它是在字节计数之后错误校验之前一个正常响应里的字节数。
- 输出状态数据充填在正常响应的数据字段中。每个字节包含 8 个输出点的值。第一个字节的最低有效位（LSB）包含输出点的值，该输出点编号等于开始点编号加上 1。输出点的值按序排列，以数据字段的第一个字节的最低 有效位开始，以数据段的最后一个字节的最高有效位结束。如果点数不是 8 的倍数，最后一个数据字节最高的 1 到 7 位是 0。

信息(02): 读输出表

格式:



查询:

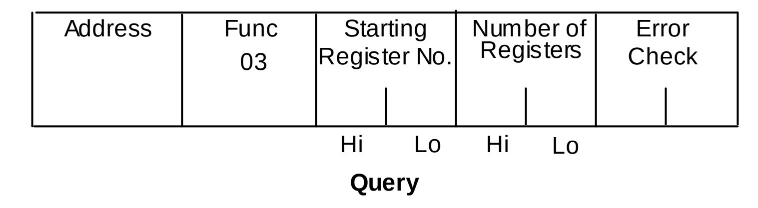
- 不允许使用地址 0，因为这不能是一个广播请求。
- 功能代码是 02.
- 开始点编号是双字节长，可以是任何小于附属 CPU 里最高输出点有效编号的值。开始点编号等于一个比第一个输出点编号小 1 的数，这个输出点是对该请求的正常响应送回的。
- 点数值是双字节长，指定一个正常响应送回的输出点数。开始点值及点数值的总和必须小于或等于附属 CPU 里最高输出点数变量的值。开始点编号的高位字节和字节字段数作为第一个字节被发送。低位字节是第二个字节。

响应:

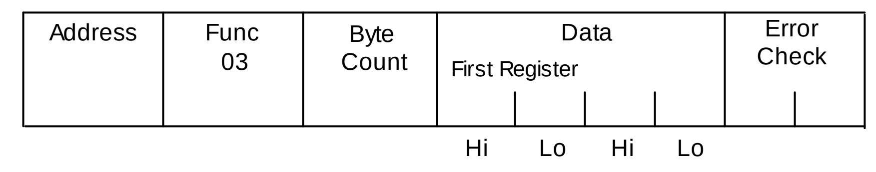
- 字节计数是一个从 1 到 256（0=256）的二进制数。它是在字节计数之后错误校验之前一个正常响应里的字节数。
- 输出状态数据充填在正常响应的数据字段中。每个字节包含 8 个输出点的值。第一个字节的最低有效位（LSB）包含输出点的值，该输出点编号等于开始点编号加上 1。输出点的值按序排列，以数据字段的第一个字节的最低 有效位开始，以数据段的最后一个字节的最高有效位结束。如果点数不是 8 的倍数，最后一个数据字节最高的 1 到 7 位是 0。

信息(03):读寄存器

格式:



Query



Normal Response

查询:

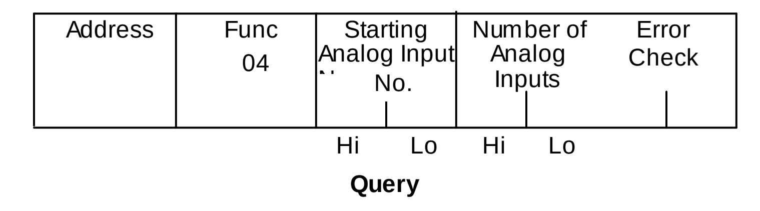
- 不允许使用地址 0，因为这不能是一个广播请求。
- 功能代码是 03.
- 开始寄存器编号是双字节长，可以是任何小于附属 CPU 里最高寄存器有效编号的值。开始寄存器数等于第一个寄存器编号加 1 的数，这个寄存器编号是由对该请求的正常响应送回的。
- 寄存器数量是双字节长，在 1 到 125 范围内（包括 1 和 125）。开始寄存器编号及寄存器数量的总和必须小于或等于附属 CPU 里最高有效输出点编号。开始寄存器编号的高位字节和寄存器数量字段作为第一个字节被发送。低位字节是第二个字节。

响应:

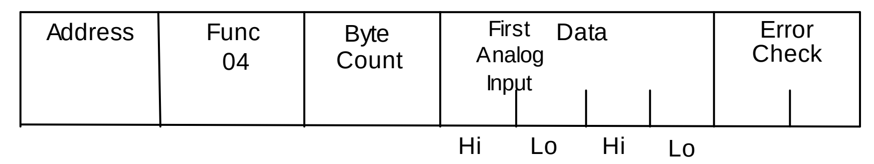
- 字节计数是一个从 2 到 250（包括 2 和 250）的二进制数。它是在字节计数之后错误校验之前一个正常响应里的字节数。注意，字节计数等于 2 倍的在一个响应里送回的寄存器数。最大设置为 250 个字节（125 个寄存器），以便整个响应适合一个 256 个字节的数据块。
- 寄存器的编号在一个数据字段里被送回，该数据字段是按照最低位编号寄存器在数据字段前两个字节，最高编号寄存器在数据字段最后两个字节的顺序排序的。在数据字段里的第一个寄存器等于开始寄存器编号加上 1。每个寄存器的高位字节在低位字节前发送。

信息(04):读模拟量输入

格式:



Query



Normal Response

查询:

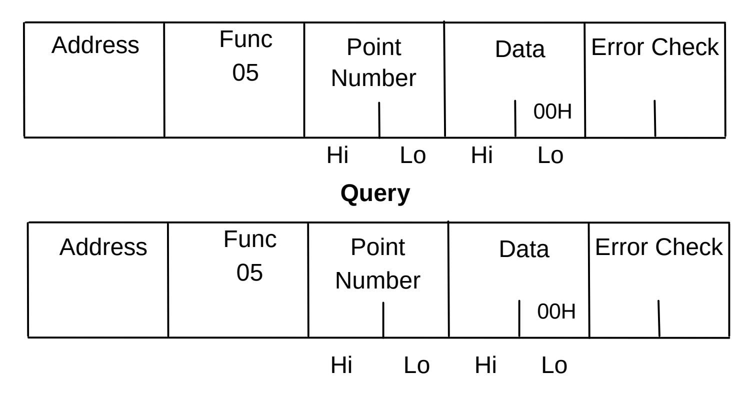
- 不允许使用地址 0，因为这不能是一个广播请求。
- 功能代码是 04.
- 模拟量输入的开始编号是双字节，可以是任何小于附属 CPU 里最高模拟量输入有效编号。模拟量输入编号等于第一个模拟量输入编号加 1，这个模拟量输入编号是对该请求的正常响应送回的。
- 模拟量输入编号值是双字节长，在 1 到 125 范围内（包括 1 和 125）。模拟量输入开始编号及模拟量输入数量的总和必须小于或等于附属 CPU 里最高模拟量输入有效编号。开始模拟量输入编号的高位字节和模拟量输入数字段作为第一个字节被发送。低位字节是第二个字节。

响应:

- 字节计数是一个从 2 到 250（包括 2 和 250）的二进制数。它是在字节计数之后错误校验之前一个正常响应里的字节数。注意，字节计数等于 2 倍的在一个响应里送回的模拟量输入数。最大设置为 250 个字节（125 个模拟量输入），以便整个响应适合一个 256 个字节的数据块。
- 送回数据字段的模拟量输入是按照最低位编号寄存器在数据字段前两个字节，最高编号寄存器在数据字段最后两个字节的顺序排列的。在数据字段里的第一个模拟量输入等于开始模拟量输入编号加上 1。每个模拟量输入的高位字节在低位字节前发送。

信息 (05):强制单个输出

格式:



查询:

- 不允许使用地址 0，因为这不能是一个广播请求。
- 功能代码是 05.
- 点编号字段是双字节长，可以是任何在附属 CPU 里最高输出点有效编号值。它等于被强制为 ON 或 OFF 的输出点数减 1。
- 数据字段的第一个字节等于 0 或 255（FFH）。如果第一个数据字段字节是 0，在点编号字段里指定的输出点被强制为 off。如果第一个数据字段字节是 FFH，在点数字段里指定的输出点被强制为 on。数据字段的第二个字节总是 0。

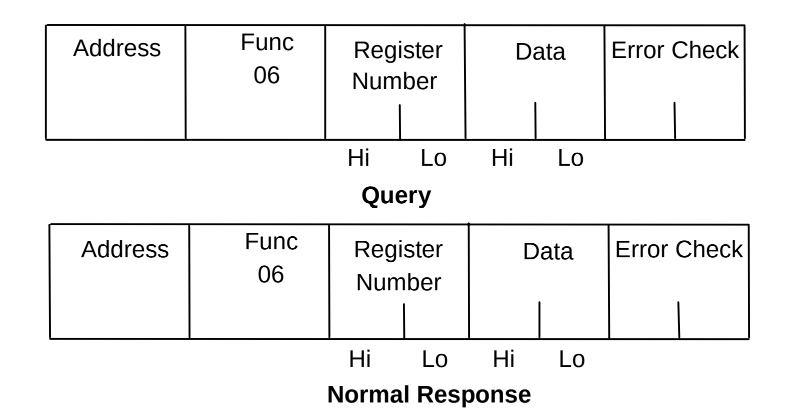
响应:

- 对一个强制单个输出查询的响应是和对查询的响应一样的。

注意: 强制单个输出请求不是一个输出越过命令。在这个请求中指定的输出确保被强制为用户逻辑扫描开始指定的值。

信息 (06):预设单个寄存器

格式:



查询:

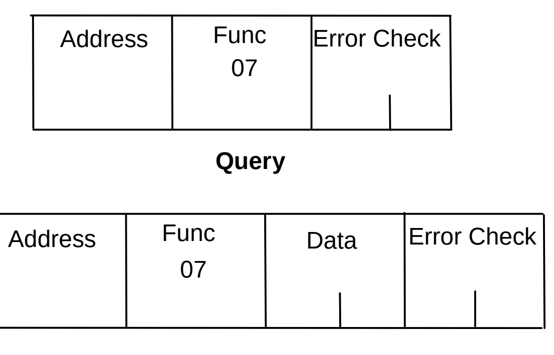
- 地址 0 表示一个广播请求。所有从站处理一个广播请求且不发送响应。
- 功能代码是 06.
- 寄存器编号字段是双字节长，可以是任何在附属 CPU 里最高寄存器有效编号值。它等于被预设寄存器数减 1 的数。
- 数据字段是双字节长，由寄存器的值组成，该寄存器由被预设的寄存器编号字段指定。数据字段的第一个字节由预设值的高位字节组成，第二个字由含低位字节组成。

响应:

- 对一个预设单个寄存器查询的正常响应是和对查询的响应一样的。

信息 (07):读例外状态

格式:



查询:

这个查询是一个为了读最初 8 个输出点的请求的简易格式。

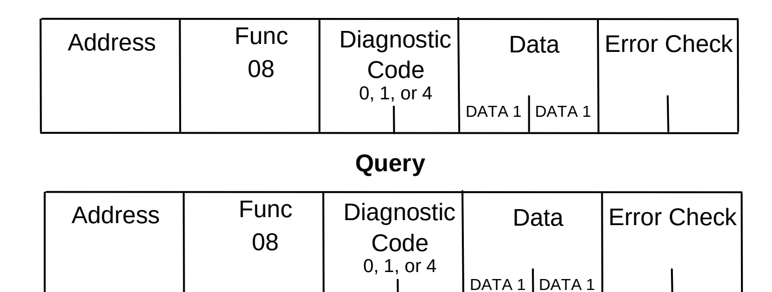
- 不允许使用地址 0，因为这不能是一个广播请求。
- 功能代码是 07.

响应:

- 正常响应的数据字段是一个字节长，由输出点 1 到 8 的状态组成。输出状态以一定顺序排列，输出点 1 的状态在最低有效位，输出点 8 的值在最高有效位。

信息(08): 回送/维护 (General)

格式:



查询:

- 功能代码是 8。
- 诊断代码是双字节长。在诊断代码字段里，诊断代码字段的高位字节是发送的第一个字节，低位字节是第二个字节。回送/维护命令只为是 0、1、4 的诊断代码定义。其他的诊断代码被保留。
- 数据字段是双字节长。两个数据字节的内容由诊断代码的值来定义。

响应:

- 见有关各个诊断代码的描述。

诊断返回查询数据请求 (回送/维护代码 00):

- 地址 0 不允许用于诊断返回查询数据请求。
- 在查询数据中两个数据字段的值是随机的。
- 对该请求的的正常响应是和对查询的响应一样的
- 在响应里数据字节的值和发送到查询的值是相等的。

诊断启动通讯重启请求回送/维护代码 01):

- 地址 0 表示一个广播请求。所有从站处理一个广播请求且不发送响应。
- 这个请求不激活只听模式（当查询被接受到，激活被发送的响应以便通讯能重启）。
- 数据字段第一个字节（DATA1）的值必须是 0 或 FF。其他值会导致发送一个错误的响应。数据字段第二个字节（DATA2）的值总是 0。
- 对开始通讯重启请求的正常响应和查询的正常响应是一样的。

诊断强制只听模式请求（回送/维护代码 04）:

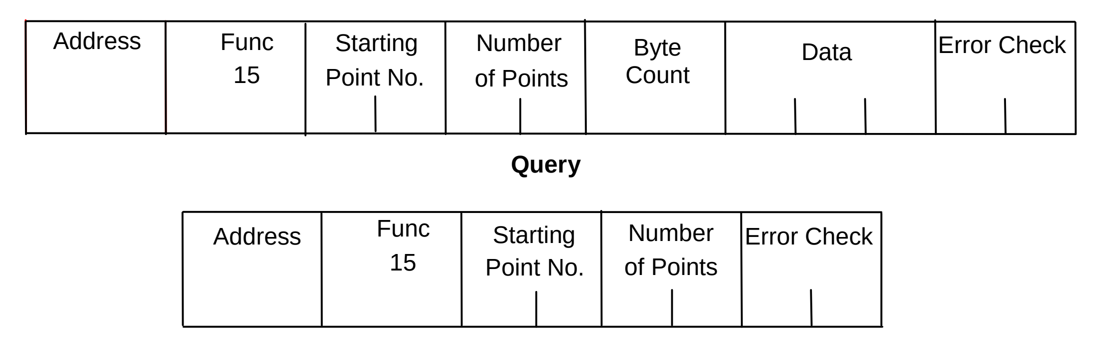
- 地址 0 表示一个广播请求。所有从站处理一个广播请求。

- 接收到一个强制只听模式请求后，RTU 设备将进入只听模式，将不执行被请求的功能，也不给查询发送正常或错误的响应。只听模式在 RTU 接收一个启动通讯重启请求或 RTU 设备上电时不激活。
- 一个强制只听模式的数据字段的两个字节都是 0。RTU 设备永远不会给一个强制只听模式请求发送一个响应。

注意: 上电时, RTU 设备不激活只听模式，被激活后连续给查询发送响应。

信息(15):强制多个输出

格式:



Query

Normal Response

查询:

- 地址 0 表示一个广播请求。所有从站处理一个广播请求且不发送响应。
- 功能代码是 15
- 开始点编号是两字节长，可以是任何小于附属 CPU 里有效的最高输出点编号。开始点编号等于一个比被该请求强制的第一个输出点编号小 1 的数。
- 点数有两个字节长。开始点编号与点数的总和必须小于附属 CPU 里有效的最高输出点编号。开始点编号的高字节和字节数字段作为这些字段的第一个字节被发送。开始点编号的低字节作为这些字段的第二个字节被发送。
- 字节计数采用一个 1 到 256 的二进制数(0 = 256)表示。它是在强制多输出点请求的数据字段中的字节数。
- 数据字段由包含应该被强制的输出点（由开始点编号和点数字段指定）的值的数据组成。在数据组中每个字节由 8 个输出点该被强制的值组成。第一个字节的最低有效位由输出点应该被强制的值组成，这个输出点的编号等于开始点编号加 1。输出点值按序排列，以数据组第一个字节的 LSB 开始，以数据字段最后一个字节的 MSB 结束。如果点数不是 8 的倍数，那么最后一个数据字节的最高的 1 到 7 位添 0。

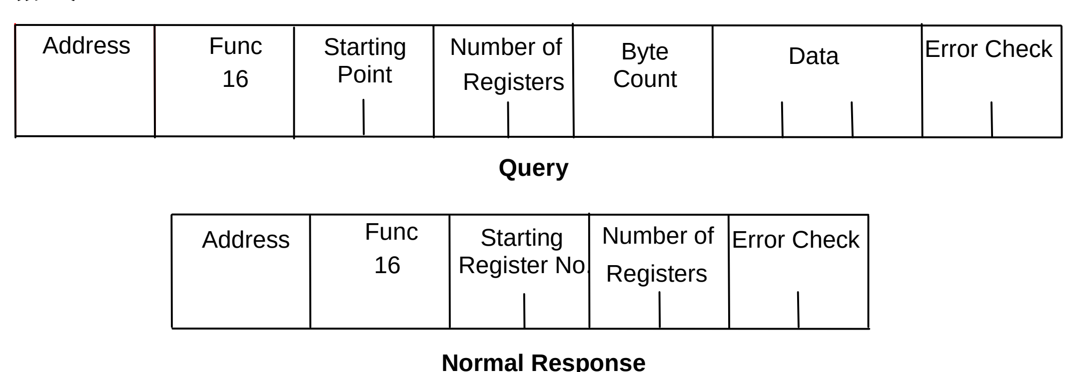
响应:

- 在响应中字段的描述和查询中字段的描述相同。.

注意: 强制多个输出点请求不是一个输出超越命令。请求中的输出确保是被强制的由用户逻辑一次扫描的开始被指定的值。

信息 (16): 预设多个寄存器

格式:



查询:

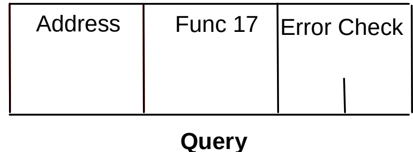
- 地址 0 表示一个广播请求。所有从站处理一个广播请求且不发送响应。
- 功能代码是 16.
- 开始寄存器的编号是两字节长，可以是任何小于附属 CPU 里有效的最高寄存器编号。开始寄存器编号等于一个比被该请求预设的第一个寄存器编号小 1 的数。
- 寄存器数量的大小用两个字节表示，在 1 到 125 范围内（包括 1 和 125）。开始寄存器的编号与寄存器数量的大小的总和必须小于或等于附属 CPU 里有效的最高寄存器编号。开始寄存器编号的高字节和寄存器数量字段作为这些字段的第一个字节被发送。开始寄存器编号的低字节作为这些字段的第二个字节被发送。
- 字节计数用一个字节表示，是一个 2 到 250 的二进制数(包括 2 和 250)。它等于在预设多个寄存器请求的数据组里的字节数。注意，字节计数等于两倍的寄存数量。
- 寄存器按照最小编号寄存器在数据组头两个字节最高编号在数据字段的最后两个字节的顺序送进数据字段。数据字段中第一个寄存器的编号等于开始寄存器加 1。每个寄存器高位字节比低位字节先发送。

响应:

- 在响应中字段的描述和查询中字段的描述相同。.

信息 (17): 报告设备类型

格式:



Query

Address	Func 17	Byte Count	Device Type 43	Slave Run Light	Data	Error Check

Normal Response

查询:

报告设备类型查询是从主设备发到从设备，目的是为了弄清楚可编程控制器或其他计算机的类型。

- 该请求不允许使用地址 0，因为这不是一个广播请求。
- 功能代码是 17.

响应:

- 字节计数字段是一个字节长，等于 5。
- 设备类型字段是一个字节长，在 PACSystems 中等于 43（十六进制）。
- 从设备运行灯字段是一个字节长，如果 CPU 在运行模式，设备运行灯字节等于 0FFH。如果 CPU 不在运行模式，设备运行灯字节等于 0。
- 数据字段包含 3 个字节。在 PACSystems CPU 中，第一个字节是次要类型，其余的字节是 0。次要类型列于下表：

响应数据 (次要类型)	*设备类型描述*
02	RX7i 300Mhz CPU (IC698CPE010)
04	RX7i 700Mhz CPU (IC698CPE020)
05	RX7i 700Mhz Redundant CPU (IC698CRE020)
06	RX7i 600Mhz CPU (IC698CPE030)
08	RX7i 1.6Ghz CPU (IC698CPE040)
0A	RX3i 300Mhz CPU (IC695CPU310)

信息(22):屏蔽写 4x 存储器

用 AND 屏蔽、OR 屏蔽和寄存器的当前内容修改指定 4x 寄存器的内容。该功能可以用来置或清除寄存器中个别位。不支持广播。

查询

查询指定 4x 参考地址被写，指定数据被用作 AND 屏蔽，指定数据被用作 OR 屏蔽。

算法如下：

结果 = (Current Contents AND And_Mask) OR (Or_Mask AND And_Mask)

例如,

	Hex	Binary	
当前内容	12	0001	0010
And_Mask	F2	1111	0010
Or_Mask	25	0010	0101
And_Mask	0D	0000	1101
结果	17	0001	0111

注意: 如果 Or_Mask 值是 0, 结果是当前内容和 And_Mask 的逻辑与，如果 And_Mask 值是 0，结果等于 Or_Mask 的值。

注意: 寄存器的内容可以用 Read Holding Register（功能代码 3）来读。只要控制器扫描用户逻辑程序，寄存器的内容随后可以被改变。

使用上面的屏蔽值，对从设备 17 寄存器 5 屏蔽写的例子:

字段名称	例 (Hex)
从地址	11
功能	16
参考地址 Hi	00
参考地址 Lo	04
And_Mask Hi	00
And_Mask Lo	F2
Or_Mask Hi	00
Or_Mask Lo	25
错误校验 (LRC 或 CRC)	--

响应

正常响应是查询的一个回声。寄存器被写后响应就被送回。

信息 (23): 读写 4x 存储器

在一个单 Modbus 事务中，完成一个读和写操作的组合。该功能可以写新的内容到一组 4x 寄存器里。不支持广播。

查询

查询指定被读的寄存器组的开始地址和数量。也指定被写的寄存器组的开始地址、数量和数据。字节计数字段指定字节数量跟在写数据字段后面。

这里是一个查询的例子，在从设备 17 里，读 6 个开始于寄存器 5 的寄存器，写 3 个开始于寄存器 16 的寄存器。：

字段名称	***例 (Hex)***
从设备地址	11
功能	17
读参考地址 Hi	00
读参考地址 Lo	04
读的数量 Hi	00
读的数量 Lo	06
写参考地址 Hi	00
写参考地址 Lo	0F
写的数量 Hi	00
写的数量 Lo	03
字节计数	06
写数据 1 Hi	00
写数据 1 Lo	FF
写数据 2 Hi	00
写数据 2 Lo	FF
写数据 3 Hi	00
写数据 3 Lo	FF
错误校验 (LRC 或 CRC)	--

响应

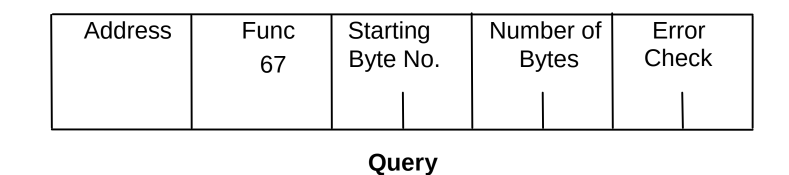
正常的响应由来自被读的寄存器组的数据组成。在读数据字段里，字节计数字段指定跟随的字节数量

这是一个查询响应的例子。：

字段名称	*例(Hex)*
从设备地址	11
功能	17
字节计数	0C
读数据 1 Hi	00
读数据 1 Lo	FE
读数据 2 Hi	0A
读数据 2 Lo	CD
读数据 3 Hi	00
读数据 3 Lo	01
读数据 4 Hi	00
读数据 4 Lo	03
读数据 5 Hi	00
读数据 5 Lo	0D
读数据 6 Hi	00
读数据 6 Lo	FF
错误校验 (LRC 或 CRC)	--

信息（67）；读缓冲存储器

格式：



Query

Address	Func 67	Byte Count	Data	Error Check

Normal Response

查询：

- 不允许使用地址 0，因为这不能是一个广播请求。
- 功能代码是 67。
- 开始字节编号在长度上是两个字节长，并可以是任何小于或等于下表说明的附属 CPU 中有效的最高缓冲寄存器地址。开始字节编号等于对这个请求的正常响应返回的第一个缓冲寄存器字节的地址。
- 字节数大小是 2 个字节长。它指定正常响应返回的缓冲寄存器存储单元（字节）的数目。开始字节编号和字节数值大小的总和必须小于 2 加上在附属 CPU 中有效的最高缓冲寄存器地址。开始字节编号和字节字段数的高位字节作为每个字段的第一个字节发送。低位字节作为每个字段的第二个字节。

响应

- 字节计数是从 1 到 256(0 = 256)的二进制数。它是正常响应中的数据字段的字节数目。
- 数据字段包括由查询请求的缓冲寄存器的内容。缓冲寄存器字节按地址顺序发送。地址等于开始字节编号的缓冲寄存器字节的内容在数据字段的第一个字节中发送。地址等于一个比开始字节编号下和字节数的大小总和小 1 的缓冲寄存器字节在数据字段的最后字节中发送。

RTU 缓冲寄存区

每次 PACSystems RTU 从设备接收到一个外部 READ 请求时，整个缓冲寄存区被更新。所有的缓冲寄存区存储单元都是只读的。缓冲寄存区是一个基于字节的存储器类型。

RTU 缓冲寄存器分配

SP 地址	字段标识符	位							
		7	6	5	4	3	2	1	0
00	CPU 运行状态	0	0	0	0	见注意 1			
01	CPU 命令状态	与 SP(00)相同的位模式							
02 03	CPU 类型	主要类型 [2a]（十六进制） 次要类型 [2b]（十六进制）							
04 – 0B	CPU SNP ID	7 位 ASCII 字符+终止符(00h)							
0C 0D	CPU 固件修订版本	主要类型(BCD) 次要类型(BCD)							
0E 0F	.通信管理模块（CMM）固件修订版本	主要类型 次要类型							
10—11	保留	00h							
12	节点类型标识符	PACSystems 43 (十六进制)							
13—15	保留	00h							
16	RTU 局地址	1—247 (十进制)							
17	保留	00h							
18—33	存储器类型 [3] 大小								
18—1B	寄存器存储器	%R 大小（字）							
1C—1F	模拟输入表	%AI 大小（字）							
20—23	模拟输出表	%AO 大小（字）							
24—27	输入表	%I 大小（位）							
		%O 大小（位）							
2C—2F	内部离散存储器	%M 大小（位）							
30—33	用户程序编码	逻辑程序占用的程序存储器的数量							
34—FF	保留	00h							

缓冲存储器分配注释

[1] 0000 = Run_Enabled 0100 = Halted
0001 = Run_Disabled 0101 = Suspended
0010 = Stopped 0110 = Stopped_IO_Enabled

[2a] CPU 主要类型代码:
PACSystems 0x43

[2b] PACSystems Minor Types for CPU:
见信息 (17) 报告设备类型

[3] 缓冲存储区字节 18h-33h：
四个字节控制带保留用于以后扩展的最高有效位的每个存储器类型的十六进制长度。例如，默认大小为 1024（0400h）字的寄存器存储器将以下面的格式返回：

字	最低	有效	最高	有效
SP 字节	18	19	1A	1B

包含	00	04	00	00

通讯错误

串行连接通讯错误被分成三组：

- 无效查询信息
- 串行连接超时
- 无效事务

无效查询信息

当通信模块接收到一个访问它本身的查询，但不能处理该查询时，就发出下面错误响应中的一个：

	子码
无效功能代码	1
无效地址字段	2
无效数据字段	3
查询处理失败	4

对一个查询的错误响应格式如下：

Address	Exception Func	Error Subcode	Error Check

地址反映原始请求规定的地址。例外代码等于查询的功能代码加上 128。错误子码 等于 1、2、3 或 4。子码的值指出了查询不能被处理的原因。

无效功能代码错误响应(1)

用子码 1 表示的错误响应叫做一个无效功能代码错误响应。如果它接收到一个功能代码不等于 1 到 8，15，16，17 或 67 的查询，这个响应就被一个从设备发送出去。

无效地址错误响应（2）

子码为 2 的错误响应叫做一个无效地址错误响应。这个错误响应在下面的情况下被发送：

1. 开始点编号和点数字段指定在附属 CPU 中无效的输出点或输入点（返回功能代码 1，2，15）。
2. 开始寄存器编号和寄存器数字段指定在附属 CPU 中无效的寄存器（返回功能代码 4，16）。
3. 开始模拟输入编号和模拟输入数字段指定在附属 CPU 中无效的模拟输入（返回功能代码 3）。
4. 点编号字段指定在附属 CPU 中无效的一个输出点（返回功能代码 5）。

5. 寄存器编号字段指定一个在附属 CPU 中无效的寄存器（返回功能代码 6）。
6. 模拟输入编号字段指定一个在附属 CPU 中无效的输入编号（返回功能代码 3）。
7. 诊断代码不等于 0，1 或 4（返回功能代码 8）。
8. 开始字节编号和字节数字段指定一个在附属 CPU 中无效的缓冲寄存器地址（返回功能代码 67）。

无效数据数值错误响应（3）

子码为 3 的错误响应叫做一个无效数据数值响应。这个响应在下面的情况下发送：

对单个输出请求(功能代码 5)或发起通信重启请求（功能代码 8，诊断代码 1），数据字段的第一个字节不等于 0 或 255(FFh)或者数据字段的第二个字节不等于 0。对强制只听 请求(功能代码 8, 诊断代码 4)，数据字段的两个字节都不等于 0。当由存储器地址地段指定的数据长度比接收到的数据长时，该响应也会发送。

查询处理失败错误响应（4）

子码为 4 的错误响应叫做查询失败错误响应。当一个 RTU 设备接收到一个查询但不能在相关 CPU 中通信，CMM 失败时， 该 RTU 设备发送这个错误响应。

串行连接暂停

导致一个 RTU 设备暂停的唯一原因是当一个信息正在被接收时发生一个四字符周期的数据流中断。如果这种情况发生，认为该信息被终止并且不再向主设备发送响应。有几个由应该被主设备重视的从设备的特征引起的时序因素。在发送一个查询信息后，在从设备不响应请求之前，主设备应该为从设备翻转等待一个适当的时间。从设备翻转时间受控制器通讯窗口时间和 CPU 扫描时间的影响。正如 13-24 页关于 RTU 的描述。

无效事务

如果在传送期间发生一个不属于无效查询信息或串行连接暂停类别的错误，该错误就被认为是无效事务。导致一个无效事务的错误类型包括：

- Bad CRC.
- 由存储器地址字段指定的数据长度比接收的数据更长。
- 成帧或运转超限错误
- 奇偶错误

如果一个信息被从串口接收时此类别的错误发生，从设备不会返回一个错误信息；从设备忽略引入的信息，处理为好象该信息不是提供给它的。

附属程序器的 RTU 从/SNP 从操作

如果一个主 SNP，像一个程序器，开始与端口之间通信，则用 RTU 从协议配置好的端口能切换到 SNP 协议。为了能够被识别，该程序器必须使用与当前激活的 RTU 从协议相同的串行通信参数（波特率，奇偶，停止位等）。当 CPU 识别主 SNP 时，从端口中移除 RTU 从协议并安装 SNP Slave 作为活动协议。

在这种情况下安装 的 SNP 协议具有下列确定的特性：

- SNP ID 设置为空。因此 SNP 主必须使用一个 SNP 附属信息中的空 ID 。这也意味着这种能力只对点对点连接有用。
- 翻转时间设为 0ms。
- 等待暂停设置为 10s。

程序器被移除后，在 CPU 识别到以前存在一个微小的延迟（等于等待暂停）。在这个时间期间，端口上没有信息被处理。CPU 检测程序器的移除当作 SNP 从超时。通过 SNP 从协议使使用中的超时失效时，小心一点是十分重要的。

当 CPU 识别 3 程序器断开连接时，CPU 重新设置 RTU 从通讯，除非在次期间一个新的协议被配置完成。这种情况下，CPU 安装新的协议代替。

例子

1. 端口 1 正以 9600 波特运行 RTU 从协议。
2. 一个程序器被放在端口 1，程序器使用 9600 波特。
3. CPU 在端口 1 上安装 RTU 从协议，程序器正常通讯。
4. 程序器把一个新的配置存储进端口 1。新的配置为 SNP 从设置端口 4800 波特。（在端口与程序器失去通讯之前不生效）
5. 当 CPU 与程序器失去通讯时，新配置生效。

SNP 从协议

PACSystems CPU 用 SNP 从协议通过端口 1 或端口 2 可以和 ME 软件通讯。

CPU 端口 1 作为一个 RS-232 数据通讯设备(DCE)端口连接，可以直接用直通电缆连接到运行 ME 或其他 SNP 主软件的 PC 的一个串行口。在使用 CPU 端口 2 时，该端口采用 RS-485 连接，需要一个外接电源的 RS-232/ RS-485 变换器（分类号 IC690ACC901），除非 SNP 主也有一个 RS-485 端口。

PACSystems 提供 SNP 的逃脱（ *break free*）方案，以便 SNP 主不需要发出一个断开信号作为 SNP 附加顺序的一部分。如果断开信号被检测到，CPU 适当地响应，通过复位协议等待来自主设备的另一个附加顺序。

PACSystems 支持点对点连接（一主/一从）和多路连接（一主/多从）。

有关 SNP 协议的详细资料，请查阅 *Serial Communications User's Manual,* GFK-0582

永久数据包

永久数据包在建立永久数据包的 SNP 会议后保存下来。这样允许一个 SNP 主设备周期性地恢复来自一个多路连接的许多 PLC 的数据包数据，省得主设备每次连接 PLC 时不得不建立和写数据包。

能够建立的最大数据包数量是 32 个。当到达这个极限，建立另外数据包的请求被拒绝。在建立另外的数据包之前必须删除一个或更多的数据包。因为 SNP 会议之后数据包不能自动检测，这个限制将阻止数据包无节制地建立。

数据包在一个电源分合循环之后不能保存下来。

SNP 的通讯请求

PACSystems 串口 1 和 2 通常不提供主设备，也不支持 SNP 命令的 COMM_REQ 功能。这些 COMM_REQ 功能可以和配置过的提供 SNP 设备的 PCM/CMM 模块一起使用。更多信息，请查阅 *Serial Communications User's Manual*, GFK-0582。

Chapter 14 故障处理

本章描述 PAC 的故障处理系统，提供了额外故障数据的定义和纠正故障的方法。

发生某种影响系统操作和性能的错误时，控制系统报告故障。一些情况，比如丢失 I/O 模块或者丢失机架，可能会削弱 PLC 控制机器或者过程的能力。同时也会指明其他的情况，比如新的模块上线并可以使用。某些情况，比如电池电压过低信号，会通知给使用者或者作为报警传给使用者。

任何可以检测到的故障都会记录到 PLC 故障表或者 I/O 故障表中。

本章信息按如下方式组织：

Overview

PACSystems CPU 检测三类故障：

故障类别	例子
内部故障(硬件)	模块没有响应 电池电压过低的情况 存储器检测错误
外部 I/O 故障 (硬件)	机架或模块丢失 增加机架或模块 丢失 Genius I/O 模块
操作失败	通讯失败 配置失败 密码访问失败

系统对故障的响应

硬件故障要求系统关闭或者容纳故障。控制系统可以容纳 I/O 故障，但是应用程序和受控过程可能不能容纳这种故障。操作失败一般可以容纳。

故障一般有三种特征:

故障表受影响	I/O 故障表 PLC 故障表
故障动作	致命的 自诊断的 信息型的
是否可配置	可配置 不可配置

故障表

PACSystems CPU 有两种故障表， 记录 CPU 内部故障的 PLC 故障表和记录 I/O 设备产生的故障的 I/O 故障表(I/O 控制器)。更多信息见 14-4 页“故障表使用”

故障动作和故障动作配置

致命故障会使对应故障表记录故障，自诊断变量置位，并且使系统停止运行。只有致命故障并且使系统停止运行。

自诊断故障会使对应故障表记录故障，任何自诊断变量置位。信息故障只是会使对应故障表记录故障。

故障动作	*CPU 响应*
致命故障	故障表记录故障 设定故障变量 转到停止/故障模式
自诊断	故障表记录故障 设定故障变量
信息故障	故障表记录故障

硬件配置可以用于标明某些故障组的故障动作。对于这些组，可以将故障动作配置为致命的或者自诊断的。当可配置的组内发生致命故障或者自诊断故障，CPU 执行配置的故障动作而不是执行故障内部标明的故障动作。

注意: 显示在扩展的故障细节框内的信息指明了故障所定义的动作类型，但不一定是实际执行的故障动作。要确定可配置的故障组内的某一故障到底会执行什麽动作，你必须参考硬件配置设定。

可配置故障组内的故障:

故障表内显示的故障动作	信息型	自诊断型	致命的
实际执行的故障动作	信息型	自诊断型或者致命的 由硬件配置中选择的动作决定	自诊断型或者致命的 由硬件配置中选择的动作决定

不可配置故障组内的故障:

故障表内显示的故障动作	信息型	自诊断型	致命的
实际执行的故障动作	信息型	自诊断型	致命的

故障表使用

在 Logic Developer 软件中显示故障表,

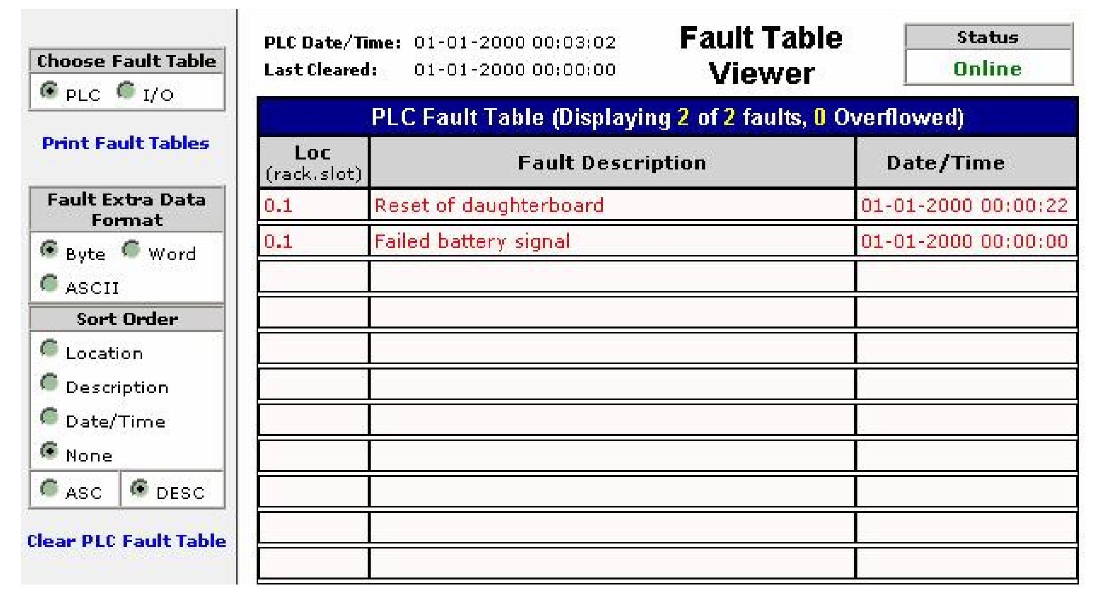
1. 与 PAC 系统同时上线
2. 选择浏览器 Navigator()内的工程(Project)键，右键单击对象(Target)节点并且选择自诊断(Diagnostics)。鼓掌表浏览器会弹出来。

PLC 故障表和 I/O 故障表会显示如下信息：

PLC 时间/日期	当前的时间和日期
最后一次清除的时间	后一次从故障表中清除故障的时间和日期。这个信息由 PLC 保存。
状态	编程器读取故障表时显示“更新(Updating)”。更新完成后状态显示为“在线(Online)”
总计故障数	最后一次清除故障表至今的故障总数。
溢出的故障数	清除故障表至今由于故障表溢出而丢失的故障数。每个故障表最多可记录 64 个故障。

PLC 故障表

PLC 故障表显示了密码侵权，配置不等，奇偶错误和通讯错误等 CPU 故障。



PLC 故障表为每个故障提供了如下信息：

位置	用机架号.槽号来标明故障位置
描述	对应于故障组，在故障细节中标明
日期时间	基于 CPU 时钟的故障日期和时间
细节	要观察日期和时间，点击故障入口。更多信息见“察看 PLC 故障细节”

察看故障细节

注意: 显示在扩展的故障细节框内的信息指明了故障所定义的动作类型，但不一定是实际执行的故障动作。要确定可配置的故障组内的某一故障到底会执行什麼动作，你必须参考硬件配置设定。

要察看 PLC 故障细节，单击故障入口。故障的详细信息会显示出来(要关闭对话框，单击这个故障)

0.1	Failed battery signal		01-02-2000 19:06:59
Error Code	Group	Action	Task Num
0	18	2:Diagnostic	0
Fault Extra Data:	02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		

PLC 故障的详细信息包含以下内容:

错误代码 进一步辨别故障。每一个故障组有自己的一套错误代码。

组 组是最高类别的故障。它标明故障的种类。编程软件显示的故障描述文本是基于故障组和错误代码的。

动作 致命的，自诊断的或者信息的。关于这些动作定义，参考 14-3 页。

任务数 大多数故障没有使用。使用时为技术支持代表提供附加信息。

额外的故障数据 为技术支持代表提供附加的可用于查找故障原因的信息。关于这些信息的解释在 14-4 页"PLC 故障描述和校正动作"中说明。

用户定义故障

用户定义故障可以记录到 PLC 故障表中。发生用户定义故障时，故障作为"应用程序消息(错误代码)"显示在适当的故障表中，并且后面有最多 24 个字符的描述性消息。用户可以定义描述性消息的所有字符。虽然消息必须以 null 字符结束，例如 0，但是 null 字符不计入 24 字符之内。如果消息中包含的字符数大于 24，则只显示 24 个字符。

特定的用户定义故障可以用于设定系统状态变量(%SA0081–%SA0112).

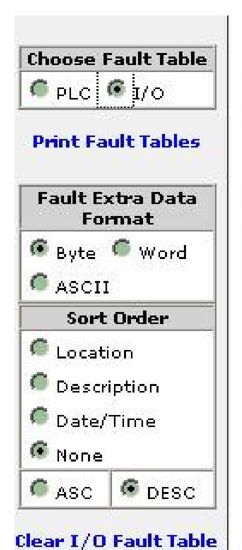
用户定义故障由服务请求 21(Service Request 21)来创建，这个服务请求在第九章描述。

注意: 户定义故障显示在 PLC 故障表中时，数值-32768 (8000 16 进制)会加到错误代码上去。例如，错误代码 5 将会显示为-32763

I/O 故障表

I/O 故障表显示 I/O 故障，比如 回路故障，地址冲突，强制的回路，I/O 模块增加/丢失和 I/O 总线故障。

故障表最多显示 64 个故障。故障表满了之后，只显示最早的 32 个故障(33—64)，和最晚发生的 32 个故障(1—32)。受到其他的故障时，第 32 个故障被移出故障表。只种情况下，前 32 个故障留给使用者察看。



PLC Date/Time: 01-01-2000 00:04:08
Last Cleared: 01-01-2000 00:00:00

Fault Table Viewer

Status: Online

I/O Fault Table (Displaying 2 of 2 faults, 0 Overflowed)

Loc	CIRC No.	Ref. Address	Fault Category	Fault Type	Date/Time
0.3	1	%AQ 00001	Circuit Fault	Analog Fault	01-01-2000 00:02:27
0.3	2	%AQ 00002	Circuit Fault	Analog Fault	01-01-2000 00:02:27

I/O 故障表为每个故障提供了如下信息：

位置 用机架号.槽号来标明故障位置，某些时候用总线和接合地址

CIRC 号码 投入使用后，标明模块的 I/O 点

变量地址* 根据对应的故障点的经验标明 I/O 存储器类型和位置(偏移地址)。当发生 Genius 设备故障或者本地模拟量模块故障时，标明的变量地址为故障模块的第一个点。

故障种类 说明故障大体的种类

故障类型 由某个故障种类的子类组成。不应用到这个种类时设为 0。

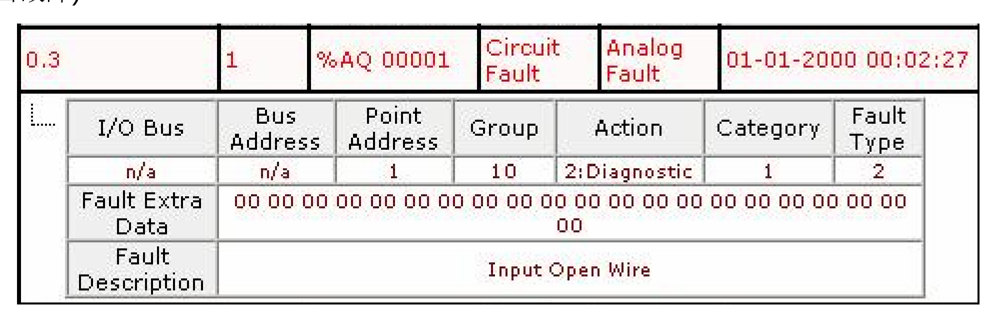
日期/时间 基于 CPU 时钟的日期和时间

细节 要察看详细信息，点击故障入口。详细信息见“察看 I/O 故障细节”

***注意:** 变量地址区域显示 16 位，而%W 范围为 32 位。偏移地址在 16 位(≤65,535)范围内的，可以正常显示。偏移地址大于 16 位的%W 变量，I/O 故障表的变量地址栏空白。

察看 I/O 故障细节

要察看 I/O 故障细节,单击故障入口。故障的详细信息对话框会显示出来。(要关闭对话框，单击故障)



0.3	1	%AQ 00001	Circuit Fault	Analog Fault	01-01-2000 00:02:27

I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type
n/a	n/a	1	10	2:Diagnostic	1	2

Fault Extra Data	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Fault Description	Input Open Wire

I/O 故障的详细信息包括:

I/O 总线	槽内的模块是一个 Genius 总线控制器(GBC)时，号码总是为 1。
总线地址	报告的或者发生故障的 Genius 设备的串行总线地址
点地址	故障为点故障时，指明发生故障的 I/O 设备上的点
组	故障组为故障的最高类别。他指明故障所属的大的种类。
动作	致命的，自诊断的或者信息的。如何确定这些动作，参考 14-3 页。
种类	指明故障种类
故障类型	用数字指明故障类型。不在这个种类上应用时设为 0
故障的额外数据	由技术支持代表提供的附加的可用于查找故障原因的信息。关于这些信息的解释在 14-33 页“I/O 故障描述和校正动作”中说明。
故障描述	I/O 故障种类为回路故障或者模块故障时提供明确的故障代码(离散回路故障，模拟回路故障，低水平模拟故障)。发生其他故障时，此值为 0。

系统故障处理

下列系统故障变量可以用于精确指定发生的故障的类型。(第七章提供完整的系统状态变量列表)

系统故障变量	*地址*	*描述*
#ANY_FLT	%SC0009	从上一次上电或者清除故障表之后两个故障表中记录的任何新的故障
#SY_FLT	%SC0010	从上一次上电或者清除故障表之后 PLC 故障表中记录的任何新的故障
#IO_FLT	%SC0011	从上一次上电或者清除故障表之后 I/O 故障表中记录的任何新的故障
#SY_PRES	%SC0012	PLC 故障表中至少有一个故障
#IO_PRES	%SC0013	I/O 故障表中至少有一个故障
#HRD_FLT	%SC0014	任何硬件故障
#SFT_FLT	%SC0015	任何软件故障

上电时，系统故障变量被清除。如果发生故障，则在故障发生后任何受影响的正向结点转换为 ON 状态。在两个故障表被清空或者整个存储器被清空前，系统故障变量一直会保持 ON 状态

系统故障变量

系统故障变量置位时，附加的故障变量也置位。这些其他类型的故障中，“可配置故障的故障变量”在下表列出，“不可配置故障的故障变量”见 14-10 页

可配置故障的故障变量

故障 (缺省动作)	地址	描述	也可以设置
#SBUS_ER (自诊断的)	%SA0032	系统总线错误。所有的系统总线故障作为信息型故障记录	#HRD_FLT, #SY_PRES, #SY_FLT
#SFT_IOC[1] (diagnostic)	%SA0029	I/O 控制器(IOC)上发生的不可恢复错误	#IO_FLT, #IO_PRES, #SFT_FLT
#LOS_RCK[2] (自诊断的)	%SA0012	丢失机架(BRM 失败，丢失电源)或者丢失一个已配置的机架	#SY_FLT, #SY_PRES, #IO_FLT, #IO_PRES
#LOS_IOC[3] (自诊断的)	%SA0013	丢失 I/O 控制器，或者丢失一个已配置的总线控制器	#IO_FLT, #IO_PRES
#LOS_IOM (自诊断的)	%SA0014	丢失 I/O 模块 (没有响应), 或者丢失一个已配置的 I/O 模块	#IO_FLT, #IO_PRES
#LOS_SIO (自诊断的)	%SA0015	丢失智能模块 (没有响应), 或者丢失一个已配置的模块	#SY_FLT, #SY_PRES
#IOC_FLT (自诊断的)	%SA0022	非致命的总线或 I/O 控制器错误，10 秒钟内超过 10 个总线错误(错误速率可配置)	#IO_FLT, #IO_PRES
#CFG_MM (致命的)	%SA0009	配置不等。检测到错误的模块类型。PLC 没有检测到为独立模块(比如 Genius I/O 模块)设定的配置参数	#SY_FLT, #SY_PRES
#OVR_TMP (自诊断的)	%SA0008	CPU 温度超过正常操作温度	#SY_FLT, #SY_PRES

1. #SFT_IOC 软件故障和你为#LOS_IOC.所设定的有同样的动作
2. 记录丢失机架或者增加机架故障时，不会产生独立的丢失或增加模块的故障。
3. 如果不是两个冗余 GBC 都发生故障，即使#LOS_IOC 故障被配置为致命故障，发生此故障时 PLC 也不会进入到停止/故障状态。

注意: 如果记录到故障表中的故障为信息型的，配置的动作不会使用。例如，如果记录的 SBUS_ERR 的故障动作为信息型的，但是你将其配置为致命的，动作仍然是信息型的。

不可配置故障的故障变量

故障	地址	描述	结果
#PS_FLT	%SA0005	电源故障	Sets #SY_FLT, #SY_PRES
#HRD_CPU (致命的)	%SA0010	CPU 硬件故障(比如存储设备失败或者串口失败)	Sets #SY_FLT, #SY_PRES, #HRD_FLT
#HRD_SIO (自诊断型)	%SA0027	系统内任何模块的非致命硬件故障	Sets #SY_FLT, #SY_PRES, #HRD_FLT
#SFT_SIO (自诊断型)	%SA0031	LAN 接口模块的不可恢复的软件错误	Sets #SY_FLT, #SY_PRES, #SFT_FLT
#PB_SUM (fatal)	%SA0001	上电过程中或运行模式下程序或块检测失败	Sets #SY_FLT, #SY_PRES
#LOW_BAT (自诊断型)	%SA0011	系统内的 CPU 或其他模块电池电压过低信号	Sets #SY_FLT, #SY_PRES
#OV_SWP (自诊断型)	%SA0002	定常扫描时间超时	Sets #SY_FLT, #SY_PRES
#SY_FULL #IO_FULL (自诊断型)	%SA0022	PLC 故障表满了(64 个). I/O 故障表满了(64 个).	Sets #SY_FLT, #SY_PRES, #IO_FLT, #IO_PRES
#APL_FLT (自诊断型)	%SA0003	应用程序故障	Sets #SY_FLT, #SY_PRES
#ADD_RCK * (自诊断型)	%SA0017	增加新机架，外部机架，或者之前发生故障的机架找到了	Sets #SY_FLT, #SY_PRES
#ADD_IOC (自诊断型)	%SA0018	外部 IOC,之前的故障 I/O 控制器不再有故障	Sets #IO_FLT, #IO_PRES
#ADD_IOM (自诊断型)	%SA0019	外部 IO 模块,之前的故障 I/O 模块不再有故障	Sets #IO_FLT, #IO_PRES
#ADD_SIO (自诊断型)	%SA0020	增加新的智能模块，或者之前有故障的模块不再有故障	Sets #SY_FLT, #SY_PRES
#IOM_FLT (自诊断型)	%SA0023	I/O 模块内的点或通道；模块的局部故障	Sets #IO_FLT, I#O_PRES
#NO_PROG (信息型)	%SB0009	上电时没有应用程序。只发生在 PLC 第一次上电或者电池为后备电源的 RAM 内包含的程序发生故障时	PLC 不会转到运行模式，在正确的程序下装到 PLC 之前，他会连续的执行停止模式扫描。这可能是不进行任何操作的空程序。设定 #SY_FLT and #SY_PRES.
#BAD_RAM (致命的)	%SB0010	上电时程序存储器崩溃。程序不可读或者不能通过检测。	Sets #SY_FLT and #SY_PRES.
#WIND_ER (信息型)	%SB0001	窗口完成错误。控制器通讯或逻辑窗口的服务被跳过。发生在定常扫描方式下。	Sets #SY_FLT and #SY_PRES.
#BAD_PWD (信息型)	%SB0011	更改权限级别的请求被拒绝，密码错误。	Sets #SY_FLT and #SY_PRES.
#NUL_CFG (致命的)	%SB0012	转换为运行模式的过程中没有配置。无配置运行相当于延缓 I/O 扫描	Sets #SY_FLT and #SY_PRES.

故障	地址	描述	结果
#SFT_CPU (致命的)	%SB0013	CPU 软件故障。CPU 检测到不可恢复错误。可能是因为看门狗时间超时	PLC 立即转换到停止/中断模式。唯一允许的活动是与编程器通讯。要清除这个故障，必须将 PLC 重新上电 。设定 SY_FLT, SY_PRES, and SFT_FLT.
#STOR_ER (致命的)	%SB0014	从编程器向 PLC 下装数据；PLC 中的某些数据可能会被清除。	PLC 不会转换到运行模式。上电时不会清除此故障，需要进行干涉才能纠正此故障。设定 SY_FLT 和 SY_PRES.

* 记录增加机架和丢失机架时，一般来讲不会生成增加或丢失的机架内部的模块的故障

使用故障结点

故障(-[F]-)和无故障 (-[NF]-)结点可以用于检测系统内目前存在的不同类型的故障。这些结点不能被覆盖。下表显示故障和无故障结点的状态。

条件	[F]	[NF]
出现故障 故障消除	ON OFF	OFF ON

故障定位变量(机架，槽，总线，模块)

PACSystems CPU 支持每个机架，槽，总线和模块的保留故障名。通过为故障和无故障结点编辑名称，逻辑可以为与机架和模块相关的故障执行相应的动作。

t 故障定位变量名称格式

这些故障名称只能在故障和无故障结点上编辑。保留的故障名总是可用的。不用启用点故障等特别的选项。

故障变量类型	保留名称	注释
机架	#RACK_000r	r 是机架号，范围 0 ~7.
槽	#SLOT_0rss	r 是机架号，范围 0 ~7. ss 是槽号, 范围 0~17.
总线	#BUS_0rssb (只适用于 Genius)	r 是机架号，范围 0 ~7， ss 是槽号, 范围 0~17, b 是 Genius 总线号
模块	#M_rssbmmm (只适用于 Genius)	r 是机架号，范围 0 ~7， ss 是槽号, 范围 0~17, b 是 Genius 总线号, and mmm 是串行总线地址(SBA)，范围 000~031.

这些故障名不与%SA, %SB, %SC 或者任何其他的变量类型对应。他们映射到一个用户不可以访问的存储器区域。只显示这些名称。

故障变量名称举例:



#RACK_0001 代表第 1 机架.

#SLOT_0105 代表第 1 机架，第 5 槽.

#BUS_02041 代表第 2 机架，第 4 槽，总线 1.

#M_2061028 代表第 2 机架，第 6 槽，总线 1, Genius 模块 28.

注释: 故障表中报告槽位失败故障时，不管槽内安装的是什麼类型的模块，与那个槽位相关的所有总线和模块故障定位变量置位(故障结点通过能流，无故障结点不通过能流)。相反的，故障表中报告槽位重启时，所有的总线和模块故障定位变量清 0。(故障结点不通过能流，无故障结点通过能流)

故障定位变量特性

上电时，PLC 内的所有故障定位变量清 0。有故障时，PLC 将受影响的变量进行状态转换。在下列动作之一发生之前，故障变量保持故障状态：

- 通过编程软件清除故障表或者清除整个存储器时，PLC 故障表和 I/O 故障表清空。
- 相关设备(机架，I/O 模块或者 Genius 设备)重新增加到系统中。故障表中记录了"Addition of. . ."故障时，PLC 会将与这个设备相关的故障变量设定到无故障状态。在另一个与这个设备相关的故障发生之前，故障变量保持无故障状态。(对于分布式 I/O 故障，特别是总线控制器重启时，这可能花费几秒钟的时间)

注意: 这些故障变量只用于信息显示。不用于验证 I/O 数据的功能。I/O 点故障变量(14-3 页描述)可能用于验证 I/O 数据。PLC 为故障状态设定故障定位变量时不会停止执行。

故障变量叠加的影响。如果第 5 机架，第 6 槽，总线 1，模块 29 发生故障，则如下故障变量会被置位：RACK_05, SLOT_0506, BUS_05061 和 M_5061029。故障表中只有 1 条故障信息来描述这个模块的故障。这种情况下，故障表不会分别描述机架，槽和总线故障。

如果一个模拟量基块(IC697ALG230)丢失，这个模块的故障变量置位。这个模块丢失后 ，他的扩展模块(IC697ALG440 和 ALG441)的故障变量不会置位。因此，扩展模块的故障定位变量置位前，会参考他的基块是否发生故障。

使用点故障

虽然也根据相关的更高级的内部硬件故障来置位，但是点故障同样适用于外部 I/O 故障(例如，IOC 故障或丢失机架)。要使用点故障，必须在 CPU 硬件配置的存储器键内将其使能。

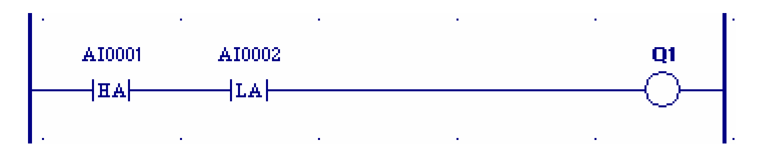
使能后，离散 I/O 的每个点和模拟 I/O 的每个通道都会在 PLC 存储器内分配。用于点故障的 PLC 存储器相当于整个变量表的大小。在 14-11 页“使用故障结点”处描述 FAULT 和 NOFLT 结点，你可以利用故障结点访问点故障。

对故障结点的完全支持依赖于 I/O 模块的能力。一些 90-30 系列模块不支持点故障结点。这些模块的点故障结点保持 off 状态，除非发生丢失模块故障，这种情况下 RX3i CPU 将这个丢失的模块的所有相关故障结点置位。

使用报警结点

高报警结点(-[HA]-)和低报警结点 (-[LA]-) 代表模拟量输入模块比较器功能的状态。要使用报警结点，必须先将 CPU 硬件配置的存储器键内将点故障功能使能。

下面的逻辑实例同时使用了高报警和低报警结点。



注意: HA 和 LA 结点产生的报警不会在故障表中记录。

PLC 故障描述和校正动作

每一个故障解释包含故障描述和校正错误的指令。很多故障描述有多个原因。这种情况下，错误代码和附加的故障信息用于区分有相同故障描述的多个故障状况。

PLC 故障组

组	名称	缺省故障动作*	是否可配置
1	丢失机架	自诊断	Yes
4	丢失可选择模块	自诊断	Yes
5	增加机架或外部机架	N/A	No
8	重启，增加或外部可选择模块	N/A	No
11	系统配置不等	Fatal	Yes
12	系统总线错误	Fatal	Yes
13	CPU 硬件故障	N/A	No
14	模块硬件故障	N/A	No
16	可选择模块软件故障	N/A	No
17	程序或模块检测错误组	N/A	No
18	电池电压过低信号组	N/A	No
19	固定扫描时间超时	N/A	No
20	系统故障表满	N/A	No
21	I/O 故障表满	N/A	No
22	用户应用程序故障	N/A	No
24	CPU 过热	自诊断	Yes
128	系统总线故障	N/A	No
129	上电时无用户程序	N/A	No
130	上电时用户程序崩溃	N/A	No
131	窗口执行失败	N/A	No
132	密码访问失败	N/A	No
134	运行模式时无系统配置	N/A	No
135	CPU 系统软件故障	N/A	No
137	存储时发生通讯故障	N/A	No
140	非关键的 CPU 软件事件	N/A	No

* 如果故障以信息方式显示，则指明的故障动作不可用。故障显示为信息型，则其特性将总是为信息型。

丢失机架(第1组)

系统不能与扩展机架进行通讯时会记录丢失机架故障，丢失机架故障可能在以下几种情况下发生：主机架的 BTM(总线发送器模块)故障，扩展机架的 BRM(总线接收器模块)故障或者在培植文件中配置了扩展机架但是上电时没有响应。

缺省动作:自诊断型。可配置

错误代码	1
名称	机架丢失
描述	主机架无法和扩展机架通讯时，PLC 发生这个错误。系统中每个扩展机架都有可能发生这种故障。
校正	(1) 系统掉电，确定 BTM 和 BRM 正确安装，并且电缆连接正确。 (2) 更换电缆 (3) 更换 BRM. (4) 更换 BTM.
错误代码	2
名称	机架无响应
描述	当从编程器下装的配置文件显示系统中有某个扩展机架但实际上那个号码的机架没有反应，则 PLC 会产生这个故障
校正	(1) 检查电源模块后面的机架号跳线-首先检查丢失的机架，再检查其他的机架-看看是否有机架号重复的现象。 (2) 如果没有此机架则需更新配置文件 (3) 如果应该有此机架，但是实际上没有，则需要在硬件配置中增加此机架。 (4) 将系统掉电，确定 BTM 和 BRM 正确安装，并且电缆连接正确。 (5) 更换电缆 (6) 更换 BRM (7) 更换 BTM. (8) 检查最后一个 BRM 的终端插头

丢失可选择模块(第4组)

LAN 接口模块，BTM 或者 BRM 没有响应时发生丢失可选择模块故障。如果上电或者存储配置时模块丢失或者在操作过程中模块没有响应，则会发生此故障。如果在通电情况下取下可选择模块，也会发生此故障。

缺省动作:自诊断型。可配置

错误代码	3C (16 进制)/60 (十进制)
名称	模块处于固件更新模式
描述	在固件更新过模式下发现模块时，PLC 会报告此故障。这种情况下模块不会与 CPU 进行通讯。
校正	(1)对这个模块进行固件更新 (2)用按钮重启这个模块 (3)将整个系统重新上电 (4)将包含这个模块的机架重新上电
错误代码	所有其他的
名称	配置过程中的模块故障
描述	上电或存储配置时模块发生故障，则 PLC 产生这个错误。
校正	(1) 将系统掉电。更换该机架的该槽内的模块 (2) 如果模块放在扩展机架上，检查 BTM/BRM 电缆连接是否紧固，校验扩展机架地址。 (3) 更换 BTM. (4) 更换 BRM. (5) 更换机架
错误代码	63(16 进制)/99 (十进制)
名称	通电情况下移走模块
描述	检测到可选模块(LAN 接口模块)在通电情况下被移走
校正	不需要校正

附加的或外部的机架 (第5组)

CPU 无法与之通讯的外部机架上线或者上电，或者发现一个没有配置的机架时产生附加的外部机架故障

动作: 不可配置

错误代码	1
名称	Extra Rack 外部机架
校正	(1) 正确设置并且上电的情况下检查机架跳线 (2) 更新配置以包含扩展机架 **注意:** 如果只是将机架上电，则不需要校正

重启，增加或者外部可选择模块(第8 组)

可选择模块(比如 LAN 接口模块，BTM 等等)上线，重启，带电插入或者发现系统内由没配置的模块时，发生重启模块，增加模块或者外部可选择模块故障。

动作: 不可配置

错误代码	8
名称	LAN 接口重启完成, 正在运行
描述	LAN 接口模块重启或者正在运行用户程序
校正	LAN 参考 LAN 接口手册，GFK-0868 或者 GFK-0869 (目前为 GFK-0533).
错误代码	7
名称	外部可选择模块
校正	(1) 更新配置以包含模块
	(2) 从系统中移走模块
错误代码	14
名称	可选择模块带电插入
描述	检测到可选择模块(比如 LAN 接口模块)带电插入时， PLC 记录此故障
校正	不需要校正

注意: 清除或者存储配置时，系统内的每个智能可选择模块自身会产生一个重启故障。

系统配置不等 (第 11 组)

模块实际所在槽位与配置文件中模块所在槽位不一致时产生配置不等故障。因为 Genius 模块而使 GBC 产生配置不等故障时，额外故障数据栏中包括不等模块的总线地址。

缺省动作:致命的。可以配置

错误代码	2
名称	Genius I/O 模块型号不等
描述	配置和 Genius I/O 模块自身的型号不等时，PLC 产生此故障
校正	(1) 更换 Genius I/O 模块，使之型号与配置文件匹配
	(2) 更新配置文件

Genius I/O 模块型号不匹配时的外部数据

字节	值
[0]	FF (标志字节)
[1]	串行总线地址
[2]	已安装的模块类型 (见 14-18页“已安装的/已配置的模块类型”.)
[3]	已配置的模块类型(见 14-18页“已安装的/已配置的模块类型”.)

已安装的/已配置的模块类型(故障外部数据的第 2，第 3 个字节)

数字		描述
十进制	十六进制	
4	4	Genius 网络接口 (GENI)
5	5	B 阶段手持监控器
6	6	B 阶段 6 系列带自诊断的 GBC
7	7	B 阶段 6 系列不带自诊断的 GBC
8	8	PLCM/6 系列
9	9	PLCM/ 90-40 系列
10	A	90-70 系列单通道总线控制器
11	B	90-70 系列双通道总线控制器
12	C	90-10 系列 Genius 通讯模块
13	D	90-30 系列 Genius 通讯模块
32	20	高速计数器
69	45	B 阶段 115Vac 8-点 (2 A) 组合模块
70	46	B 阶段 115Vac/125Vdc 8-点独立模块
70	46	B 阶段 115Vac/125Vdc 8-点独立模块不带失效转换
71	47	B 阶段 220Vac 8-点组合模块
72	48	B 阶段 24-48Vdc 16-点类似接受器模块

数字		*描述*
十进制	*十六进制*	
72	48	B 阶段 24Vdc 16-点类似接受器模块
73	49	B 阶段 24-48Vdc 16-点源块
73	49	B 阶段 24Vdc 16-16-点类似源块
74	4A	B 阶段 12-24Vdc 32-点接受器模块
75	4B	B 阶段 12-24Vdc 32-点源块
76	4C	B 阶段 12-24Vdc 32-点 5V 逻辑块
77	4D	B 阶段 115Vac 16-点嵌入状态输入块
78	4E	B 阶段 12-24Vdc 16-点嵌入状态输入块
79	4F	B 阶段 115/230Vac 16-点 常开中继块
80	50	B 阶段 115/230Vac 16-点 常闭中继块
81	51	B 阶段 115Vac 16-点 AC 输入块
82	52	B 阶段 115Vac 8-点 低泄漏组合块
127	7F*	Genius 网络适配器 (GENA)
131	83	B 阶段 115Vac 4-输入, 2-输出模拟块
132	84	B 阶段 24Vdc 4-输入, 2-输出模拟块
133	85	B 阶段 220Vac 4-输入, 2-输出模拟块
134	86	B 阶段 115Vac 热电偶输入块
135	87	B 阶段 24Vdc 热电偶输入块
136	88	B 阶段 115Vac RTD 输入块
137	89	B 阶段 24/48Vdc RTD 输入块
138	8A	B 阶段 115Vac 应变规格/mV 模拟量输入块
139	8B	B 阶段 24Vdc 应变规格/mV 模拟量输入块
140	8C	B 阶段 115Vac 4-输入, 2-输出电流源模拟量块
141	8D	B 阶段 24Vdc 4-输入, 2-输出电流源模拟量块

**GENA 应用程序 ID 号*

如果型号为 16 进制的 7F (Genius 网络适配器),模块可能为下列的一种： (GENA 应用程序 ID 作为参考量列出)

数字		
十进制	*十六进制*	*描述*
131	83	115Vac/230Vac/125Vdc 电源监控器模块
132	84	24/48Vdc 电源监控器模块
160	A0	Genius 远程 90-70 机架控制器

错误代码	4
名称	I/O 类型不等
描述	当自身的 I/O 类型和配置的 I/O 类型不等时，PLC 产生此故障

校正	(1)	移走实际的 Genius 模块，替换为配置文件中指明的模块
	(2)	更新配置中的模块描述，令其和实际安装的模块匹配

I/O 类型不匹配的额外故障数据

字节	*值*
[0]	FF
[1]	总线地址
[2]	已安装的模块的 I/O 类型
[3]	已配置的模块的 I/O 类型

已安装的 Genius 模块的 I/O 类型 (额外故障数据的第 2 个字节)

值	*描述*
01	只适用于输入
02	只适用于输出
03	混合型

已配置的 Genius 模块的 I/O 类型 (额外故障数据的第 3 个字节)

值		*描述*
十进制	*十六进制*	
0	0	离散量输入
1	1	离散量输出
2	2	模拟量输入
3	3	模拟量输出
4	4	离散量组合
5	5	模拟量组合
20	14	模拟量输入，离散量输出
21	15	模拟量输入，离散量输出
24	18	模拟量输入，离散量组合
30	1E	模拟量输出，离散量输入
31	1F	模拟量输出，离散量输出
34	22	模拟量输入，离散量组合
50	32	模拟量组合，离散量输入
51	33	模拟量组合，离散量输出
54	36	模拟量组合，离散量组合

错误代码	8
名称	模拟量扩展器不等
描述	实际的模拟量扩展器型号和配置的型号不等时 CPU 产生此错误
校正	(1) 移走实际的模拟量扩展器模块，替换为配置文件中指明的模块
	(2) 更新配置文件

错误代码	9
名称	Genius I/O 模块大小不匹配
描述	实际的模块配置大小和配置的大小不等时 CPU 产生此错误
校正	重新配置此模块

Genius I/O 模块大小不匹配的额外故障数据

字节	*值*
[0]	FF
[1]	总线地址
[2]	模块广播数据长度
[3]	已配置的模块广播数据长度

错误代码	十六进制时为 A /十进制时为 10
名称	不支持此特征
描述	改版后的模块不支持配置此特征
校正	(1) 将此模块更新为支持此特征的版本
	(2) 更改模块配置

不支持的特征的额外故障数据

字节	*值*
[8]	包含一个指明原因号码，说明不知其那些特征 0x5 – GBC 版本太老 0x6 –用在主机架时支持

错误代码	十六进制 E /十进制 14
名称	LAN 上有重复的 MAC 地址
描述	LAN 接口模块和 LAN 上的其他设备有相同的 MAC 地址。模块下线
校正	(1) 更改模块的 MAC 地址
	(2) 更改其他设备的 MAC 地址
错误代码	十六进制 F /十进制 15
名称	LAN 上重复的 MAC 地址已经处理
描述	之前的 MAC 地址重复问题已经解决。模块重新回到网络中。只作为消息显示。
校正	不要求
错误代码	十六进制 10 /十进制 16
名称	LAN MAC 地址不匹配
描述	软切换用户编制的 MAC 地址和从软件中下装的配置不一致

校正	更换软切换用户或者软件中的 MAC 地址
错误代码	十六进制 11/十进制 17
名称	LAN 软切换/Modem 不匹配
描述	LAN 的配置类型和 modem 或者软切换用户编制的配置不等
校正	(1) 纠正 modem 类型
	(2) 察看 LAN 接口手册中的配置设定部分

错误代码	十六进制 19/十进制 25
名称	DCD 长度不匹配
校正	见额外故障数据.

DCD 长度不匹配的额外故障数据

字节	*值*
[0]	FF
[1]	总线地址
[2]	模块指向的数据长度
[3]	已配置的 模块指向的数据长度

错误代码	十六进制 25/十进制 37
名称	控制器变量超出范围
描述	触发，失效或者 I/O 说明超出配置范围
校正	将不正确的变量修改值正确的范围内，或者将变量数据的配置范围扩大
错误代码	十六进制 26/十进制 38
名称	错误的程序规范
描述	程序的 I/O 规范崩溃
校正	联系 GE Fanuc 现场服务
错误代码	十六进制 27/十进制 39
名称	未定的或失效的中断变量
描述	中断触发变量超出范围或者在 I/O 模块配置中被 disabled（使之失效）时，CPU 产生这个错误。
校正	(1)移走或纠正中断触发变量
	(2)更新配置文件将这个中断使能。
错误代码	十六进制 43/十进制 67
名称	模块配置失败
描述	模块配置不被模块接收
校正	察看其他模块的故障，确定模块不接受配置的原因。检查模块配置是否正确
错误代码	十六进制 4C/十进制 76
名称	ECC 使能,但是不应该使能
描述	如果 CPU 固件不支持冗余，则 ECC 跳线不能处于使能位置
校正	将 ECC 跳线设定到位置 (跳线放在一个针脚上或者移走)

错误代码	所有其他的
名称	模块和配置不等
描述	槽内的模块类型和配置文件指示的类型不一致，这时候 CPU 会产生这个故障。
校正	(1) 根据配置文件中的描述更换模块。
	(2) 更新配置文件

系统总线错误(第12 组)

遇到总线错误时，发生系统总线错误组故障

缺省动作:自诊断。可配置

错误代码	4
名称	未知的 VME 中断源
描述	产生一个未知的中断时，CPU 会发生此故障(未配置或未知的).
校正	(1) 确定为中断配置的模块和程序逻辑中的中断声明相对应

CPU 硬件失败 (第13 组)

CPU 检测到硬件失败-比如 RAM 失败或者通讯端口失败-时发生 CPU 硬件组故障

发生 CPU 硬件失败时，OK 指示灯会闪烁以表明故障还不太严重，控制器可以找回故障信息。

动作: 不可配置

错误代码	十六进制 6E /十进制 110
名称	没有电池时的当前时间时钟
描述	会使有电池时的时钟值丢失
校正	(1) 更换电池。在更换完成之前不要将主机架掉电。使用编程软件重启当前时间时钟。 (2) 更换模块
错误代码	十六进制 0A8/十进制 168
名称	紧急的过热失败
描述	CPU 过热
错误代码	所有其他的
校正	更换模块

CPU 硬件失败的额外故障数据

对于 CPU 的 RAM 失败 (其中的一个故障报告为 CPU 硬件失败), 失败的地址存储在对话框的前四个字节中

模块硬件失败 (第 14 组)

CPU 在系统内的任何模块检测到非致命的硬件失败(比如 LAN 接口模块串口故障)时都会发生模块硬件失败故障组故障。本组的故障动作为自诊断。

动作: 不可配置

错误代码	十六进制 1A0/十进制 416
名称	丢失 12 伏电源
描述	进行 LAN 接口模块时需要 12 电源供电
校正	(1) 安装/更换 GE Fanuc 100 瓦电源 (2) 为 VME 连接外部 12 负供电电源
错误代码	1C2 - 1C6 十六进制(450 – 454 十进制)
名称	LAN 接口硬件失败
描述	关于这些错误的描述，参考 LAN 接口手册 I, GFK-0868 或者 GFK-0869 (目前为 GFK-0533)
错误代码	所有其他的
名称	模块硬件失败
描述	检测到 模块硬件失败
校正	更换受影响的模块

可选择模块软件故障 (第16组)

下列情况发生时，会有可选择模块软件故障：

- 智能可选择模块上发生不可恢复的软件故障
- 从模块上读取的数据显示模块是 GE Fanuc 模块，但是模块不是 CPU 支持的 GE Fanuc 类型
- 以太网接口在额外记录表中记录了一个事件

动作:不可配置

错误代码	1
名称	不被支持的模板类型
描述	从模块上读取的数据显示模板是 GE Fanuc 模板，但是模板不是 CPU 支持的 GE Fanuc 类型
校正	(1) 更新配置文件并检查关于这块板子的软件设定。如果有错误，将其纠正，并将纠正后的配置文件下装，再试一下。 (2) 在编程器上显示 PLC 故障表。联系 GE Fanuc 现场服务 ，为他们提供故障记录中的所有信息。
错误代码	2, 3
名称	COMMREQ 频率过高
描述	将 COMM_REQ 向模块传输的速率太快，模块无法处理。
校正	将受影响的模块的 COMM_REQ 的传输速率降低，或者在传输下一个 COMM_REQ 之前监控这个 COMM_REQ 传输的完成状况。
错误代码	4
名称	机架中不止一个 BTM
描述	机架中目前有不止一个 BTM
校正	从机架中移走一个 BTM; CPU 机架上只能有一个 BTM
错误代码	>4
名称	可选择模块软件故障
描述	在可选择模块上检测到软件故障
校正	(1) 将软件重新下装到指示的模块中 (2) 更换模块
错误代码	>4
名称	LAN 系统软件故障
描述	以太网接口软件检测到不正常状况并将其记录到额外的记录中。以太网额外记录中的额外故障数据包含相应的事件，也可以通过以太网接口的站管理其功能浏览。额外故障数据的前 2 位位时间类型；剩余的数据对应入口 2 到入口 6 的 4 位数值。一些额外记录也可能包括可选择的多字节 SCode 和其他的数据。
校正	关于外部故障的信息，参考 PAC 系统 *TCP/IP* 通讯站管理器手册，GFK-2225, 附录 B.

程序或块检测错误(第17组)

当CPU在接收的块中检测到错误状况时发生程序或块检测失败故障。运行模式后台检测时也会发生此故障。所有情况下，PLC故障表的额外故障数据中记录了发生错误的程序或者块的名称

动作:不可配置

错误代码	所有的
名称	程序或块检测失败
描述	程序或块崩溃时CPU产生这个错误
校正	(1) 清除CPU存储器并且重新存储 (2) 检查C应用程序，看是否有错误 (3) 在编程器上显示PLC故障表。联系GE Fanuc现场服务，为他们提供故障记录中的所有信息。

额外的故障数据

额外的故障数据中的前8个字节包含受影响的程序块的名称。

电池电压过低信号 (第18组)

CPU检测到CPU板，CPU存储器子板或者LAN接口模块上的电池电压过低信号时，发生电池电压过低故障。

动作: 不可配置

错误代码	0
名称	电池信号失败
描述	CPU模块 (或其他有电池的模块) 电池死掉
校正	更换电池。在换好电池之前，不要讲机架断电。
错误代码	1
名称	电池电压过低信号
描述	CPU或其他模块上的电池电压过低
校正	更换电池。在换好电池之前，不要讲机架断电。

固定扫描时间超时(第19组)

固定扫描模式时的 CPU 操作检测到固定扫描定时器时间超时时，发生固定扫描时间超时故障。额外故障数据用八个字节描述文件夹的名称。

动作: 不可配置

错误代码	0, 固定扫描 1, 没有使用
校正	固定扫描模式下: (1) 增加固定扫描时间 (2) 从应用程序中移走部分逻辑

系统故障表满(第20组)

PLC 故障表达到限制时发生系统故障表满故障(见 3-4 页)

动作: 不可配置

错误代码	0
校正	清空 PLC 故障表

I/O 故障表满(第21组)

I/O 故障表达到限制时发生 I/O 故障表满故障(见 3-6 页)。为避免附加故障丢失，清除故障表中记录的较早时发生的故障。

动作: 不可配置

错误代码	0
校正	清除 I/O 故障表

用户应用程序故障 (第 22 组)

CPU 在用户程序中检测到故障时，发生应用程序故障。

动作: 不可配置

错误代码	2
名称	软件看门狗定时器超时
描述	看门狗定时器超时时，CPU 产生这个错误。CPU 停止执行用户程序并且进入到停止/中断模式。唯一的恢复方法是将 CPU 电池取下，然后重新给 CPU 上电。引起定时器超时的可能为回环，跳转，非常长的程序等等。
校正	(1) 确定是什麼(逻辑执行，外部事件等等)引起超时并将其纠正
	(2) 使用系统服务功能块重启看门狗定时器。

错误代码	7
名称	应用程序堆栈溢出
描述	块调用的深度超出 CPU 能力
校正	增加程序堆栈大小或者减少程序的嵌套。

错误代码	十六进制 11 十进制 17
名称	程序运行时发生错误
描述	执行程序的过程中发生运行错误
校正	纠正应用程序的相关问题

错误代码	十六进制 22 十进制 34
名称	不支持的协议
描述	硬件不支持配置的协议

错误代码	十六进制 33 十进制 51
名称	读取闪存失败
描述	可能的原因:
	1. 文件没有在闪存中。(可能是因为向闪存中写入数据过程中重新上电引起的)
	2. 因为启动了 OEM 保护而不能从闪存中读取数据

错误代码	十六进制 34 十进制 52
名称	存储器变量超出范围
描述	在逻辑执行时使用的用户逻辑存储器变量超出范围。包括非直接变量，排列元素变量和潜在的其他类型的变量。
校正	纠正逻辑或者在硬件配置调整存储器的大小

错误代码	十六进制 35 十进制 53
名称	零除错误
描述	用户程序中包括零除操作
校正	纠正逻辑

错误代码	十六进制 36 十进制 54
名称	操作数不是字节排列
描述	对于要求的操作，用户逻辑中的变量不是正确的字节排列。
校正	纠正逻辑或者在硬件配置调整存储器的大小

CPU 过热 (第 24 组)

缺省动作: 自诊断。可配置

错误代码	1
名称	过热失败
描述	超出 CPU 的正常操作温度
校正	关掉 CPU 散热并且安装风扇来调节温度。

电源故障(第 25 组)

动作: 不可配置

错误代码	1
名称	供电失败
描述	未知的电源失败
校正	更换电源模块
错误代码	2
名称	电源过载
描述	电源达到负载上限
校正	更换更大容量的电源或者重新配置系统来降低耗电量
错误代码	3
名称	电源关掉
描述	电源上的开关切换的 OFF 位置
错误代码	4
名称	电源超过正常操作温度
描述	电源温度离自动关断温度只差几度
校正	关掉系统散热。安装风扇调节温度。

上电时无用户程序(第 129 组)

CPU 上电过程中，PLC 存储器内无用户程序时发生此故障。CPU 上电时检测到没有用户程序；控制器处于停止模式。

动作: 不可配置

校正	在试图转到运行模式前下装应用程序。

上电时用户程序崩溃 (第 130 组)

CPU 检测到用户 RAM 崩溃时发生此故障。. CPU 保持在停止状态。

动作: 不可配置

错误代码	1
名称	上电时用户程序崩溃
描述	在上电过程中检测到用户 RAM 崩溃时发生此错误
校正	(0) 在没有电池时重新上电
	(2) 检查 C 程序，看是否有错误
	(3) 更换 CPU 电池
	(4) 更换 CPU 上的扩展存储器板.
	(5) 更换 CPU

窗口不能完成 (第 131 组)

窗口不能完成故障由 PLC 中之前的逻辑和完成扫描过程软件产生。错误发生时，额外故障数据中包含正在执行的任务名

动作: 不可配置

错误代码	0
名称	窗口不能完成
描述	在编程器窗口有机会开始执行之前，固定扫描模式下的固定扫描时间超时会导致 CPU 产生这个故障。
校正	增加固定扫描时间值
错误代码	1
名称	逻辑窗口被跳过
描述	因为没有时间执行，逻辑窗口被跳过
校正	(1) 增加基本循环时间
	(2) 减少通讯窗口时间

密码访问失败 (第 132 组)

CPU 接收到改变权限等级的请求，但是用户输入的密码不是目标等级的密码，这种情况下会产生实际密码失败错误。

动作: 不可配置

错误代码	0
校正	使用正确密码重新尝试

运行模式下没有系统配置 (第 134 组)

CPU 从停止模式转换到运行模式的过程中发现没有配置文件时发生此故障。可以转换为运行模式，但是不能进行 I/O 扫描。

动作: 信息型。不可配置

错误代码	0
校正	下装配置文件

CPU 系统软件故障(第 135 组)

此故障由 CPU 操作软件产生。他们在系统操作的多个不同点产生。发生致命故障时，CPU 立即切换到停止/中断状态。在此种模式下，CPU 唯一可以执行的就是和编程器通讯。清除这种情况的方法是在不安装电池的情况下重新上电。

动作: 不可配置

错误代码	十六进制 5A 十进制 90
名称	要求用户断电
描述	当应用程序中执行 SVCREQ #13 (用户断电)时，CPU 产生这个信息型报警
校正	不需要。只是报警信息
错误代码	
名称	
描述	
校正	
错误代码	十六进制 94 十进制 148
名称	单元中包含的固件不等，建议更新
描述	冗余状态变化并且冗余 CPU 包含不兼容固件时记录此故障
校正	确保冗余 CPU 中有兼容固件
错误代码	十六进制 DA/十进制 218
名称	紧急过热失败
描述	CPU 紧急过热
校正	关掉 CPU 散热。安装风扇调节温度。
错误代码	所有其他的
名称	CPU 内部系统错误
描述	不应该在生产系统中发生的系统内部错误

校正	在编程器上打开 PLC 故障表。联系 GE Fanuc 现场服务，为他们提供故障记录中的所有信息。

存储过程中的通讯失败 (第 137 组)

程序，块和其它数据存储到 PLC 时可能发生存储过程中的通讯失败故障。存储的程序，块和数据的命令/数据流有独特的开始和结束顺序。如果编程器进行存储时发生意外中断而结束存储过程，就会记录故障。只要这种故障在系统中出现，控制器就不能切换到运行模式。

上电时，这个故障不会自动清除，你必须单独清除这个故障。

动作: 不可配置

错误代码	0
校正	清除故障，重新下装程序或配置文件
错误代码	1
描述	运行模式存储时，通讯丢失或者电源丢失。新的程序或者块不会激活并且会被删除。
校正	重新进行运行模式存储。这个故障时自诊断型的。
错误代码	2
描述	运行模式存储时，通讯丢失或者清除旧程序/旧块时电源丢失。新程序/新块安装进去，旧程序/旧块被删除
校正	不需要。此故障为信息型。
错误代码	3
描述	运行模式存储过程中电源丢失。
校正	删除或重新存储程序。这个错误是致命的。

不关键的 CPU 软件事件(第 140 组)

本组用于记录对技术支持有价值的系统内部状况。

缺省动作: 不可配置

错误代码	*描述*
1-30	上电过程中的事件
31-50	串口或串口协议上的事件
51 及以上	各种系统内部事件
校正	除非故障和其他某些故障同时发生，否则不需进行矫正。如果碰到其他的问题，这些信息可能有利于技术支持判断问题原因。

I/O 故障描述和校正动作

I/O 故障表会报告故障的下列相关数据:

- 故障组
- 故障动作
- 故障种类
- 故障类型
- 故障描述

所有的故障都会有个种类，但是不会将每个故障的故障类型和故障组都列出来。要观察故障的详细信息，单击 I/O 故障表的故障入口

注意　型号不等和 I/O 类型不等故障在系统配置的 PLC 故障表中记录。不会在 I/O 故障表中报告。

额外故障数据

I/O 故障表中包含最多 21 个字节的 I/O 故障额外数据，其中包含和这个故障相关的附加信息。不是所有的故障入口都包含 I/O 额外故障数据。

I/O 故障组

组号	组名	缺省故障动作*	是否可配置
2	丢失 IOC	自诊断	Yes
3	丢失 I/O 模块	自诊断	Yes
6	增加外部 IOC	N/A	No
7	增加外部 I/O 模块	N/A	No
9	IOC 或者 I/O 总线故障 Bus Fault	自诊断	Yes
10	I/O 模块故障	N/A	No
15	IOC 软件故障	与第 2 组相同 **	Yes
16	模块软件故障	N/A	No
133	Genius 块地址不匹配	N/A	No

* 如果故障显示为信息型的，则指明故障动作不可用。故障显示为信息型，就总为信息型。

** IOC 软件失败组(第 15 组)的故障动作总是和丢失 IOC 组(第 2 组)动作一致。如果配置了丢失 IOC 组，IOC 软件失败组总是配置为采用相同故障动作

I/O 故障种类

种类	*故障类型*	*故障描述*	*额外故障数据*
回路故障(1)	离散故障(1)	用户侧电源丢失(01 十六进制)	回路故障
		用户接线短路(02 十六进制)	回路故障
		持续过流(04 十六进制)	回路故障
		电流过低或没有电流(08 十六进制)	回路故障
		转换温度过高(10 十六进制)	回路故障
		转换失败 (20 十六进制)	回路故障
		点故障 (83 十六进制)	回路故障
		数出熔断器熔毁 (84 十六进制)	回路故障
	模拟故障(2)	输入通道低报警(01 十六进制)	回路故障
		输入通道高报警(02 十六进制)	回路故障
		输入通道低于下限(04 十六进制)	回路故障
		输入通道高于上限 (08 十六进制)	回路故障
		输入通道开环 (10 十六进制)	回路故障
		高于上限或开环 (18 十六进制)	回路故障
		输入通道低于下限(20 十六进制)	回路故障
		输入通道高于上限(40 十六进制)	回路故障
		扩展通道没有响应(80 十六进制)	回路故障
		数据不正确(81 十六进制)	回路故障
	低级 模拟故障(4)	输入通道低报警(01 十六进制)	回路故障
		输入通道高报警(02 十六进制)	回路故障
		输入通道低于下限(04 十六进制)	回路故障
		输入通道高于上限 (08 十六进制)	回路故障
		输入通道开环 (10 十六进制)	回路故障
		接线错误 (20 十六进制)	回路故障
		内部故障(40 十六进制)	回路故障
		输入通道短路 (80 十六进制)	回路故障
		数据不正确(81 十六进制)	回路故障
	GENA (Genius 网络适配器) 故障 (3)	GENA 回路故障 (80 十六进制)	字节 2:GENA 故障
丢失模块 (2)	没指定(0) A/D 通讯丢失(1)	NA	模块配置 输入回路号 输出回路号
增加模块(3)	NA	NA	模块配置 输入回路号 输出回路号
I/O 总线故障(6)	总线故障(1) 总线输出 Disabled (2) SBA 冲突 (3)	NA	NA

种类	*故障类型*	*故障 描述*	*额外故障数据*
Genius 模块故障 (8)	数据转发器故障(0) A 到 D 通讯故障(1) 用户缩放错误 (5) 存储失败 (6)	配置存储器失败(08 十六进制) 标定配置存储器失败(20 十六进制) 共享的 RAM 失败 Failure (40 十六进制) 内部回路故障 (80 十六进制) 看门狗时间超时 (81 十六进制) 输入熔断器熔毁(84 十六进制)	NA

种类	*故障类型*	*故障* *描述*	*额外故障数据*
增加 IOC (9)	NA	外部模块 (01 十六进制) 重启请求 (02 十六进制)	NA
丢失 IOC (10)	NA	NA	超时 非预期的状态 r 非预期的邮件状态 VME 总线错误
IOC 软件故障 (11)	NA	NA	NA
强制回路(12)	NA	NA	块配置 离散/模拟说明*
未强制回路(13)	NA	NA	块配置 离散/模拟说明*
丢失 I/O 模块 (14)	NA	NA	NA
增加 I/O 模块(15)	NA	VME 模块重启请求(30 十六进制)	NA
额外 I/O 模块(16)	NA	NA	NA
额外块(17)	NA	NA	NA
IOC 硬件故障 (18)	NA	NA	NA
因为故障太多，GBC 停止报告故障(19)	GBC 检测到 Genius 总线上错误过多，会下线至少 1.5 秒钟(1)	NA	NA
GBC 软件意外 (21)	数据表队列满了 (1) R/W 请求队列满了 (2) 低优先级的邮件被拒绝 (3) CPU 完成初始化之前接受到后台消息(4) Genius 软件版本太旧(5) 过多使用 GBC 内存 (6)	NA	
块转换(22) – 冗余 Genius 块转换总线	NA	NA	块配置 输入回路数 输出回路数 移走模块的 GBC 机架/槽号地址
冗余总线上的块没有激活(23)	NA	NA	NA

回路故障(第1类)

回路故障应用到 Genius I/O 模块和 IC697VRD008 RTD/Strain 桥接模块。这一类的所有故障都有额外故障数据。具体到某一个故障，会有不止一种信息报告出来。

动作: 自诊断

回路故障的额外故障数据

总线控制器

回路故障入口使用额外故障数据的 1 个或者 2 个字节。如果 GBC 报告故障，第一个字节由 GBC 产生，第二个字节按如下方式编码：.

树值 (字节 2)	*描述*
1	输入回路
2	输入回路
3	输出回路

如果故障类型为 GENA 故障,第 2 个字节包含了从 GENA 模块报告来的“故障报告”消息的第 2 个字节的数据。

VRD001 RTD/Strain 桥

回路故障有 13 个字节的额外故障数据区域。额外故障数据按下表编制：

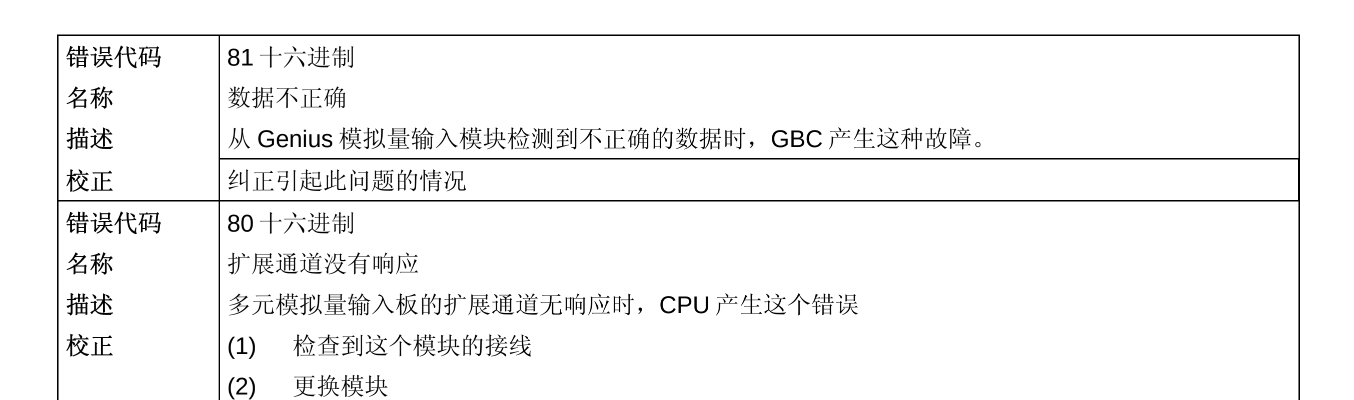
字节	*描述*
1--10	由技术支持使用
11	线号
12	模块号
13	由技术支持使用

离散故障的故障描述

错误代码	1
名称	用户侧电源丢失
描述	Genius I/O 模块现场接线端掉电后，GBC 产生这个错误
校正	(1) (只适用于独立 I/O 模块.)初始化 “脉冲测试” COMREQ #1。脉冲测试可以在 I/O 模块中使能或取消。
	(2) 纠正电源失败
错误代码	2
名称	用户接线短路
描述	检测到 Genius 模块用户接线短路时，GBC 产生这个错误。电流超过 20 安培时认为短路。
校正	修复短路故障
错误代码	4
名称	持续过流
描述	检测到用户接线端电流持续超过 2 安培时，GBC 产生这个故障。
校正	修复过流故障
错误代码	8
名称	低电流或无电流
描述	用户回路电流非常低或者无电流时 GBC 产生这种错误。
校正	修复这种情况
错误代码	10 十六进制
名称	转换开关温度过高
描述	Genius 智能转换开关温度过高时，GBC 产生这种错误。
校正	(1) 确保安装的模块有足够的流量
	(2) 降低模块环境温度
	(3) 在 RC 感应负载上安装缓冲器
错误代码	20 十六进制
名称	转换失败
描述	Genius 智能转换开关转换失败时，GBC 产生这种错误。
校正	(1) 检查 Genius 输出旁路(按钮).
	(2) 更换 Genius I/O 模块
错误代码	83 十六进制
名称	点故障
描述	检测到 Genius I/O 模块上单个 I/O 点失败时，CPU 产生这种错误
校正	更换 Genius I/O 模块
错误代码	84 十六进制
名称	输出熔断器熔断
描述	检测到 Genius I/O 输出模块熔断器熔断时,CPU 产生这种错误。
校正	(1) 确定并且修理熔断故障；更换熔断器
	(2) 更换模块

模拟故障的故障描述

错误代码 名称 描述 校正	1 输入通道低报警 Genius 模拟量模块报告输入通道有低报警时，GBC 产生这个错误 纠正引起低报警的情况
错误代码 名称 描述 校正	2 输入通道高报警 Genius 模拟量模块报告输入通道有高报警时，GBC 产生这个错误 纠正引起高报警的情况
错误代码 名称 描述 校正	4 输入通道在最低限幅之下 Genius 模拟量模块报告输入通道值在最低限幅之下时，GBC 产生这个错误 纠正引起此问题的情况
错误代码 名称 描述 校正	8 输入通道在最高限幅之上 Genius 模拟量模块报告输入通道值在最高限幅之上时，GBC 产生这个错误 纠正引起此问题的情况
错误代码 名称 描述 校正	10 十六进制 输入通道开环 Genius 模拟量模块检测到输入通道开环时，GBC 产生这个错误 纠正引起此问题的情况
错误代码 名称 描述 校正	18 十六进制 最高限幅之上或者开环 输入开环或者输入不可缩放 纠正引起此问题的情况
错误代码 名称 描述 校正	20 十六进制 输出通道在最低限幅之下 Genius 模拟量模块报告输出通道值在最低限幅之下时，GBC 产生这个错误 纠正引起此问题的情况
错误代码 名称 描述 校正	40 十六进制 输出通道在最高限幅之上 Genius 模拟量模块报告输出通道值在最高限幅之上时，GBC 产生这个错误 纠正引起此问题的情况



错误代码	81 十六进制
名称	数据不正确
描述	从 Genius 模拟量输入模块检测到不正确的数据时，GBC 产生这种故障。
校正	纠正引起此问题的情况
错误代码	80 十六进制
名称	扩展通道没有响应
描述	多元模拟量输入板的扩展通道无响应时，CPU 产生这个错误
校正	(1) 检查到这个模块的接线 (2) 更换模块

低级模拟量错误

错误代码	1
名称	输入通道低报警
描述	Genius 模拟量模块报告输入通道有低报警时，GBC 产生这个错误
校正	纠正引起低报警的情况
错误代码	2
名称	输入通道高报警
描述	Genius 模拟量模块报告输入通道有高报警时，GBC 产生这个错误
校正	纠正引起高报警的情况
错误代码	4
名称	输入通道在最低限幅之下
描述	Genius 模拟量模块报告输入通道值在最低限幅之下时，GBC 产生这个错误
校正	纠正引起此问题的情况
错误代码	8
名称	输入通道在最高限幅之上
描述	Genius 模拟量模块报告输入通道值在最高限幅之上时，GBC 产生这个错误
校正	纠正引起此问题的情况
错误代码	10 十六进制
名称	输入通道开环
描述	Genius 模拟量模块检测到输入通道开环时，GBC 产生这个错误
校正	纠正引起此问题的情况
错误代码	20 十六进制
名称	接线错误
描述	Genius 模拟量模块检测到不正确的 RTD 连接或者热电偶转换连接故障时，GBC 产生这个错误
校正	纠正引起此问题的情况
错误代码	40 十六进制
名称	内部故障
描述	Genius 模拟量模块报告热电偶模块上的冷接合传感器错误或者 RTD 模块内部故障时，GBC 产生这个错误
校正	纠正引起此问题的情况
错误代码	80 十六进制
名称	输入通道短路
描述	检测到 Genius RTD 或者应变规格块有短路状况时，GBC 产生这个错误。
校正	纠正引起此问题的情况
错误代码	81 十六进制
名称	不正确的数据
描述	检测到 Genius 模拟量输入模块上有不正确的数据时，GBC 产生这个错误
校正	纠正引起此问题的情况

GENA 故障

GENA 故障没有故障描述。 GENA 故障的字节 2 是额外故障数据的第 1 个字节。

错误代码	80 十六进制
描述	检测到和 Genius I/O 总线连接的 GENA 模块失败时，Genius I/O 操作软件产生这个错误
校正	更换 GENA 模块.

丢失模块(第 2 类)

适用于 Genius 设备的丢失模块类故障

动作: 自诊断型

名称	丢失模块
描述	不能与 Genius 设备通讯时，GBC 产生这个错误
校正	(1) 校验模块的电源和接线 (2) 更换模块
名称	丢失模块 - A/D 通讯故障 Communications Fault
描述	检测到 Genius 设备上的 A/D 通讯失败时，GBC 产生这个错误
校正	(1) 校验电源和到模块的串行接线 (2) 更换模块

丢失模块的额外故障数据

丢失模块故障有 4 个字节的额外故障数据。第 2 个字节包含模块配置并且按下表编码。第 3 个字节说明了可能使用的输入回路号，第 4 个字节说明了可能使用的输出回路号。

模块配置 (第 2 字节)

值	*描述 Description*
1	模块被配置为输入模块
2	模块被配置为输出模块
3	模块被配置为输入输出模块(组合模块)

增加模块 (第 3 类)

增加模块这一类故障只适用于 Genius 设备 。这个种类的故障没有故障类型和故障描述。

动作: 自诊断

名称	增加模块
描述	检测到已经停止与控制器通讯的 Genius 块重新与控制器进行通讯时，Genius 操作软件产生这个错误。
校正	只是信息型的。不需要

增加模块的额外故障数据

增加模块故障有 4 个字节的额外故障数据。第 2 个字节包含模块配置并且按下表编码。第 3 个字节说明了可能使用的输入回路号，第 4 个字节说明了可能使用的输出回路号。

模块配置 (第 2 字节)

值	*描述*
1	模块被配置为输入模块
2	模块被配置为输出模块
3	模块被配置为输入输出模块(组合模块)

I/O 总线故障(第 6 类)

I/O 总线故障有三种故障类型：

缺省动作: 自诊断。可配置

名称	总线故障
描述	检测到 Genius I/O 总线失败时，GBC 操作软件产生这个错误。 (超过 GBC 配置中的错误速率时产生这个错误—缺省值为 10，意味着 10 秒钟 10 个错误 t).
校正	(1) 确定总线失败的原因并且纠正过来。 (2) 更换 GBC. (3) 缺省值为 10。如果需要，可以设得更高。但是应该用示波器检测总线。
名称	总线输出 Disabled
描述	当等待 CPU 进行输出扫描的时间超时，GBC 操作软件产生这个错误。
校正	(1) 将其设为扫描设定 1，以减少 GBC 输出扫描时间间隔。 (2) 增加 CPU 软件看门狗时间设定 (3) 更换 CPU. (4) 在编程器上显示 PLC 故障表。联系 GE Fanuc 技术支持，为他们提供故障表中的相关信息。
名称	SBA 冲突
描述	GBC 检测到自己的串行总线地址和总线上其他设备的地址冲突
校正	将冲突的串行总线地址中的一个进行调整

模块故障(第8类)

模块故障类有一种故障类型，数据转发故障和 8 个故障描述。这类故障没有额外故障数据。缺省故障动作为自诊断。

错误代码	08 十六进制
名称	配置存储器失败
描述	检测到 Genius 模块的 EEPROM 或者 NVRAM 失败时，GBC 产生这个错误
校正	更换 Genius 模块的电子模块
错误代码	20 十六进制
名称	校准存储器失败
描述	检测到 Genius 模块的校准存储器失败时，GBC 产生这个错误
校正	更换 Genius 模块的电子模块
错误代码	40 十六进制
名称	共享 RAM 故障
描述	检测到 Genius 模块的共享 RAM 失败时，GBC 产生这个错误
校正	更换 Genius 模块的电子模块
错误代码	81 十六进制
名称	看门狗时间超时
描述	检测到输入模块看门狗时间超时时，CPU 产生这个错误
校正	更换输入模块
错误代码	80 十六进制
名称	模块故障
描述	检测到模块内部失败
校正	更换受影响的模块
错误代码	84 十六进制
名称	输出熔断器熔断
描述	检测到输出模块熔断器熔断故障时，CPU 产生这个错误。
校正	(1) 确定和修复熔断器熔断原因，并且更换熔断器 (2) 更换模块

增加IOC (第9类)

增加 I/O 控制器故障类没有故障类型和相关的故障描述。本类故障的缺省故障动作为自诊断.

名称	增加 IOC
描述	故障 IOC 重新回到系统中，配置文件中没有、但是在系统中发现 IOC 或者 IOC 热插入时，CPU 产生这个错误。
校正	(1) 如果故障模块在远端机架，由于远端机架重新上电而使模块重新回到系统中，则不需要进行动作。 (2) 更新配置文件或者移走模块。
错误代码	01 十六进制
名称	额外模块
描述	模块出现，但是没有配置这个模块
校正	更新配置文件或者移走模块
错误代码	02 十六进制
名称	重启请求
描述	在重启请求之后，模块重新回到系统
校正	不需要进行动作

丢失IOC (第10类)

丢失 I/O 控制器故障类没有故障类型和相关的故障描述。

缺省动作: 自诊断。可配置

注意: 不管配置的动作是什麽，在 I/O 故障表中，这个故障总是显示为致命的

名称	丢失 IOC
描述	不能与 I/O 控制器进行通讯或者不能与配置文件中的 IOC 入口进行通讯时，CPU 产生这种故障。 IOC 在带电情况下被移走也会记录此故障。(这种情况下不需进行纠正)
校正	(1) 确定槽位/总线地址中的模块是正确的模块。 (2) 浏览配置文件，确定配置文件正确。 (3) 更换模块 (4) 在编程器上显示 PLC 故障表。联系 GE Fanuc 技术支持，为他们提供故障表中的相关信息。

丢失IOC的额外故障数据

丢失 IOC 的额外故障数据为技术支持提供了查找故障的附加信息

IOC (I/O 控制器) 软件故障 (第 11 类)

IOC 软件故障类适用于任何类型的 I/O 控制器

动作: 致命的

名称 描述 校正	数据表队列满，读/写队列满 太多的数据表或者读/写请求送到 GBC. 调整系统，减少发给 GBC 请求速率.
名称 描述 校正	响应丢失 GBC 不能对收到的数据表和读/写请求作出响应 调整系统，减少发给 GBC 请求速率.

强制和非强制回路 (第 12 和 13 类)

描述	强制和非强制回路报告的是点的状况，因此不是技术故障。他们没有故障类型和故障描述。通过手持监控器强制或取消强制 Genius I/O 点时，产生这些报告。 动作: 信息型

强制和非强制回路的额外故障数据

增加或移除回路强制时会产生 3 个字节的额外故障数据：

字节号	描述	值	描述
1	回路配置	1	回路为输入回路
		2	回路为输入回路
		3	回路为输出回路
2	模拟/离散信息	1	模块为离散块
		2	模块为模拟块
		3	模块为离散/模拟混合块

丢失 I/O 模块 (第 14 类)

丢失 I/O 模块故障适用于离散和模拟 I/O 模块。这类故障没有故障类型和相关的故障描述.

缺省动作: 自诊断。可配置.

名称	丢失 I/O 模块
描述	检测到 I/O 模块不再响应 CPU 的命令或者配置文件显示某个槽位应该有 I/O 模块但实际上没有时，CPU 产生这个错误。I/O 模块被带电移走时也会产生此故障 (这种情况下不需要纠正动作)
校正	(1) 更换模块 (2) 纠正配置文件 (3) 在编程器上显示 PLC 故障表。联系 GE Fanuc 技术支持，为他们提供故障表中的相关信息。

增加 I/O 模块 (第 15 类)

增加 I/O 模块故障应用于离散和模拟 I/O 模块。这类故障没有故障类型和相关的故障描述.

动作: 自诊断

名称	增加 I/O 模块
描述	故障的 I/O 模块返回系统中或者带电插回系统中时，CPU 产生这种错误。
校正	(1) 模块被移走或者更换或者远端机架被重新上电时不需要有动作。 (2) 更新配置文件或者移走模块
错误代码	30 十六进制
名称	VME 根据请求重启
描述	需要重启 VME 模块
校正	不需要进行动作

额外 I/O 模块(第 16 类)

额外 I/O 模块故障应用于离散和模拟 I/O 模块。这类故障没有故障类型和相关的故障描述.

动作: 自诊断

名称	额外 I/O 模块
描述	配置文件中没有此模块，但是槽内有模块时，CPU 产生这个错误
校正	(1) 移走模块 (可能是槽位错误) (2) 更新配置文件并且将包括额外模块的的配置文件重新下装

额外模块 (第 17 类)

额外模块故障只应用于 Genius I/O 设备。这类故障没有故障类型和相关的故障描述.

动作: 自诊断

名称	额外模块
描述	串行总线上有模块，但是配置文件中的没有此模块时，GBC 产生这个错误
校正	(1) 移走或者重新配置此模块 (串行总线地址可能是错误的) (2) 更新配置文件并且将包括额外模块的的配置文件重新下装

IOC 硬件失败(第 18 类)

IOC 硬件失败故障没有故障类型和相关的故障描述.

动作: 自诊断

描述	总线通讯硬件或者波特率不匹配时，发生硬件失败故障。检测到此故障时，Genius 操作软件产生此错误。
校正	(1) 确定配置文件中的 GBC 波特率和总线上每个模块的波特率匹配。 (2) 需要的话更新配置文件并且重新下装配置文件 (3) 更换 GBC. (4) 有选择的逐个移走模块，并且替换有问题的模块

GBC 停止报告故障(第 19 类)

描述	GBC 检测到 Genius I/O 总线错误率过高，并且离线至少 1.5 秒
校正	检查不正确的接线，其他设备的干扰，连接松动或者 Genius 总线上的不正常的设备

GBC 软件意外 (第 21 类)

名称	接收数据表满 (1)
描述	太多的数据表或者读/写请求发给 GBC.
校正	调整系统降低对 GBC 的请求速率.
名称	读/写请求序列满(2)
描述	GBC 内的读/写请求序列满请求可能来自 Genius 总线或者 COMM_REQ
校正	调整系统降低对 GBC 的请求速率.
名称	从 GBC 到 CPU 的低等级邮件队列满 (3)
描述	传给 CPU 的响应丢失
名称	CPU 完成初始化(4)之前收到要求 CPU 进行动作的 Genius 后台消息。
描述	消息被忽略
名称	GBC 软件版本太老 (5)
校正	更新 GBC 固件
名称	过多使用 GBC 内存 (6)
校正	校验 COMMREQ 的使用

模块转换 (第 22 类)

模块转换故障没有故障类型和相关的故障描述.

动作: 自诊断

名称	模块转换
描述	冗余 Genius 总线上的 Genius 模块从一个总线转换到另一个总线时，GBC 产生这个故障
校正	(1) 不需要进行动作来保持模块操作 (2) 转换之前模块所在的总线可能需要修复 (a)校验总线接线 (b)更换 I/O 控制器 (c)更换总线转换模块 (BSM).

模块转换的额外故障数据

字节号	*描述*	*值*	*描述*
1	回路配置	1	回路为输入回路
		2	回路为输入回路
		3	回路为输出回路
2	模块配置	1	模块配置为只用于输入
		2	模块配置为只用于输出
		3	模块配置为用于输入输出组合模块
3	使用的输入回路号		
4	使用的输出回路号		

Appendix A 性能数据

本附录包含每一种 PAC 系统 CPU 模块的指令和定时器信息。定时器信息可以用于预测 CPU 扫描时间。本附录内的信息按如下编制：

注意: 在手册印刷以后，所有 CPU 的所有数据可能不再可靠。附加的信息会增添到后续的版本

布尔执行时间

典型布尔执行时间	
IC695CPU310	每 1000 个布尔结点/线圈 0.195ms
IC698CPE010	每 1000 个布尔结点/线圈 0.195ms
IC698CPE020	每 1000 个布尔结点/线圈 0.14ms
IC698CRE020	每 1000 个布尔结点/线圈 0.14ms

指令定时

本部分的表中列出了 PAC CPU 执行每个功能所需时间的微秒数。下表数据是对 CPU310 version 2.8, CPE010 version 2.57, CPE020 version 2.57 和 CRE020 version 2.04 做测试得出的。

每个功能列出了 2 种执行时间。不同长度的输入增加的时间说明如下：

执行时间	描述
Enabled	输入正确，通过能流时函数或函数块的执行时间
Disabled	非 enabled 时所需的执行时间
Increment	由于输入长度而增加到函数基本执行时间上的时间。只适用于可以使用不同长度输入参数的函数(Search, Array Moves 等等) ■ 对于表功能，增加值 (increment)以长度为单位 ■ 对于位操作函数，increment 为毫秒/位 ■ 对于数据移动函数，微秒/单位

注意:

- 所有时间为典型执行时间。输入变化或者发生错误时时间会发生变化。
- Enabled 时间为字存储器单个单位的时间
- COMMREQ 时间是 NOWAIT 选项下测试的 CPU 和以太网模块通讯时间
- DOIO 时间是用离散输出模块测试的
- 每次扫描时间期满后定时器更新

	CPU310			CPE010			CPE020			CRE020		
Function	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***
定时器												
ONDTR	6.8	5.4	0	6.8	5.4	0	2.9	2.3	0	2.7	2.2	0
OFDT	6.3	5.7	0	6.0	5.4	0	2.6	2.3	0	2.7	2.4	0
TMR	6.4	6.0	0	6.4	5.7	0	2.7	2.4	0	2.6	2.4	0
Counters												
UPCTR	5.8	5.8	0	6.1	6.1	0	2.6	2.6	0	2.5	2.5	0
DNCTR	5.9	5.9	0	6.0	6.0	0	2.5	2.5	0	2.4	2.4	0
数学函数												
ADD (INT)	3.5	1.3	0	3.4	1.6	0	1.5	0.7	0	1.5	0.6	0
ADD (DINT)	3.7	1.3	0	3.7	1.6	0	1.6	0.7	0	1.6	0.6	0
ADD_REAL	3.6	1.3	0	3.7	1.6	0	1.6	0.7	0	1.5	0.6	0
SUB (INT)	3.6	1.4	0	3.5	1.7	0	1.5	0.7	0	1.6	0.6	0
SUB (DINT)	3.6	1.3	0	3.5	1.6	0	1.5	0.7	0	1.6	0.6	0
SUB_REAL	3.6	1.3	0	3.6	1.6	0	1.5	0.7	0	1.5	0.6	0
MUL (INT)	3.6	1.3	0	3.5	1.6	0	1.5	0.7	0	1.6	0.6	0
MUL (DINT)	3.7	1.3	0	3.6	1.7	0	1.5	0.7	0	1.5	0.6	0
MUL MIXED	3.6	1.4	0	3.6	1.8	0	1.5	0.7	0	1.6	0.6	0
MUL_REAL	3.6	1.3	0	3.7	1.6	0	1.6	0.7	0	1.5	0.6	0
DIV (INT)	3.8	1.3	0	3.5	1.7	0	1.5	0.7	0	1.6	0.6	0
DIV (DINT)	3.9	1.3	0	3.7	1.7	0	1.6	0.7	0	1.6	0.6	0
DIV_REAL	3.6	1.3	0	3.7	1.7	0	1.6	0.7	0	1.6	0.6	0
MOD (INT)	3.6	1.4	0	3.5	1.7	0	1.5	0.7	0	1.6	0.6	0
MOD (DINT)	3.7	1.3	0	3.7	1.6	0	1.6	0.7	0	1.6	0.6	0
ABS (INT)	3.0	1.1	0	2.9	1.4	0	1.2	0.6	0	1.3	0.5	0
ABS (DINT)	3.0	1.2	0	3.1	1.5	0	1.3	0.6	0	1.4	0.6	0
ABS_REAL	3.1	1.1	0	3.0	1.5	0	1.3	0.6	0	1.3	0.5	0
SQRT (INT)	3.2	1.1	0	3.4	1.4	0	1.4	0.6	0	1.5	0.5	0
SQRT (DINT)	3.7	1.1	0	4.0	1.3	0	1.7	0.6	0	1.6	0.5	0
SQRT_REAL	3.0	1.1	0	3.1	1.4	0	1.3	0.6	0	1.3	0.5	0
三角函数												
SIN	3.3	1.1	0	3.4	1.4	0	1.5	0.6	0	1.4	0.5	0
COS	3.6	1.1	0	3.4	1.4	0	1.5	0.6	0	1.4	0.5	0
TAN	3.4	1.1	0	3.6	1.4	0	1.6	0.6	0	1.4	0.5	0
ASIN	3.6	1.1	0	3.7	1.4	0	1.6	0.6	0	1.6	0.5	0
ACOS	3.6	1.1	0	3.7	1.4	0	1.6	0.6	0	1.4	0.5	0
ATAN	3.3	1.1	0	3.4	1.4	0	1.5	0.6	0	1.7	0.5	0
对数												
LOG	3.4	1.1	0	3.7	1.6	0	1.6	0.7	0	1.4	0.5	0
LN	3.4	1.1	0	3.5	1.4	0	1.5	0.6	0	1.4	0.5	0
EXPT	5.6	1.4	0	5.3	1.7	0	3.4	0.7	0	2.2	0.6	0
EXP	3.7	1.1	0	3.9	1.4	0	1.7	0.6	0	1.2	0.5	0
关系												
EQ (INT)	4.1	1.2	0	3.9	1.5	0	1.6	0.6	0	1.8	0.6	0
EQ (DINT)	4.0	1.2	0	3.8	1.5	0	1.6	0.6	0	1.6	0.6	0
EQ_REAL	3.8	1.2	0	3.7	1.5	0	1.6	0.7	0	1.6	0.5	0
NE (INT)	4.0	1.2	0	3.7	1.5	0	1.6	0.6	0	1.7	0.5	0
NE (DINT)	3.8	1.1	0	3.7	1.5	0	1.6	0.6	0	1.6	0.5	0
NE_REAL	4.1	1.1	0	3.7	1.5	0	1.6	0.6	0	1.6	0.5	0
GT (INT)	4.1	1.1	0	3.8	1.5	0	1.6	0.6	0	1.8	0.6	0
GT (DINT)	4.0	1.2	0	3.8	1.6	0	1.6	0.7	0	1.6	0.5	0
GT_REAL	3.8	1.2	0	3.7	1.5	0	1.6	0.6	0	1.6	0.5	0

	CPU310			CPE010			CPE020			CRE020		
Function	**Enabled**	**Disabled**	**Increment**	**Enabled**	**Disabled**	**Increment**	**Enabled**	**Disabled**	**Increment**	**Enabled**	**Disabled**	**Increment**
GE (INT)	4.2	1.2	0	3.8	1.7	0	1.6	0.6	0	1.8	0.5	0
GE (DINT)	3.9	1.2	0	3.7	1.5	0	1.6	0.6	0	1.6	0.6	0
GE_REAL	3.8	1.2	0	3.7	1.6	0	1.6	0.7	0	1.6	0.5	0
LT (INT)	4.2	1.2	0	3.9	1.5	0	1.6	0.6	0	1.8	0.5	0
LT (DINT)	3.8	1.1	0	3.7	1.5	0	1.6	0.6	0	1.6	0.5	0
LT_REAL	3.8	1.1	0	3.7	1.5	0	1.6	0.6	0	1.6	0.6	0
LE (INT)	4.2	1.2	0	3.9	1.5	0	1.6	0.6	0	1.8	0.6	0
LE (DINT)	3.9	1.2	0	3.8	1.5	0	1.6	0.7	0	1.6	0.5	0
LE_REAL	3.9	1.1	0	3.7	1.5	0	1.6	0.6	0	1.6	0.7	0
CMP (INT)	5.2	1.3	0	5.3	1.5	0	2.3	0.7	0	2.3	0.6	0
CMP (DINT)	5.4	1.3	0	5.2	1.5	0	2.2	0.6	0	2.3	0.6	0
CMP_REAL	5.5	1.3	0	5.2	1.6	0	2.2	0.7	0	2.3	0.6	0
RANGE (INT)	4.8	2.1	0	5.6	2.4	0	2.4	1.0	0	2.2	1.0	0
RANGE (DINT)	5.1	2.0	0	5.0	2.3	0	2.2	1.0	0	2.2	1.0	0
RANGE (DWORD)	5.0	2.1	0	5.0	2.4	0	2.2	1.0	0	2.1	1.0	0
位操作												
AND (WORD)	5.3	1.7	0.11	4.9	2.0	0.11	2.1	0.9	0.05	2.2	0.8	0.05
AND(DWORD)	5.2	1.7	0.17	5.0	1.8	0.17	2.1	0.8	0.66	2.1	0.8	0.07
OR (WORD)	5.3	1.6	0.11	4.8	1.9	0.11	2.1	0.8	0.05	2.1	0.8	0.05
OR (DWORD)	5.6	1.7	0.17	5.4	1.9	0.17	2.3	0.8	0.07	2.2	0.8	0.07
XOR (WORD)	5.2	1.7	0.11	4.9	1.9	0.11	2.1	0.8	0.05	2.2	0.8	0.05
XOR (DWORD)	5.3	1.7	0.17	5.0	1.9	0.17	2.1	0.8	0.07	2.1	0.8	0.07
NOT (WORD)	4.3	1.4	0.09	4.0	1.7	0.09	1.7	0.7	0.04	1.7	0.6	0.04
NOT (DWORD)	4.4	1.4	0.13	4.4	1.7	0.13	1.9	0.7	0.06	1.7	0.6	0.06
MCMP (WORD)	9.1	2.7	0.23	9.0	2.8	0.22	3.9	1.2	0.10	4.0	1.1	0.10
MCMP (DWORD)	9.0	2.5	0.25	9.2	2.7	0.25	3.9	1.2	0.11	4.1	1.1	0.11
SHL (WORD)	6.6	2.6	0.17	6.6	2.8	0.17	2.8	1.2	0.07	2.9	1.2	0.07
SHL (DWORD)	6.8	2.7	0.22	6.8	3.1	0.22	2.9	1.3	0.08	2.9	1.2	0.10
SHR (WORD)	6.8	2.6	0.18	6.7	2.8	0.18	2.9	1.2	0.1	2.9	1.2	0.08
SHR (DWORD)	7.0	2.7	0.23	6.8	2.8	0.23	2.9	1.2	0.1	2.9	1.2	0.10
ROL (WORD)	4.7	1.6	0.19	4.3	1.9	0.06	1.8	0.8	0.08	1.8	0.7	0.08
ROL (DWORD)	4.7	1.6	0.20	4.5	1.9	0.20	2.0	0.8	0.09	1.9	0.8	0.09
ROR (WORD)	4.7	1.7	0.19	4.8	2.0	0.19	2.1	0.8	0.08	1.9	0.8	0.08
ROR (DWORD)	4.8	1.6	0.20	4.3	1.9	0.20	1.8	0.8	0.09	2.0	0.7	0.08
BTST (WORD)	4.4	1.5	0	4.5	1.8	0	1.9	0.8	0	2.0	0.7	0
BTST (DWORD)	4.6	1.4	0	4.5	1.7	0	1.9	0.7	0	2.0	0.7	0
BSET (WORD)	3.7	1.3	0	3.5	1.6	0	1.5	0.7	0	1.5	0.6	0
BSET (DWORD)	3.6	1.4	0	3.5	1.6	0	1.5	0.7	0	1.5	0.6	0
BCLR (WORD)	3.6	1.3	0	3.6	1.6	0	1.6	0.7	0	1.5	0.6	0
BCLR (DWORD)	3.5	1.3	0	3.6	1.6	0	1.6	0.7	0	1.5	0.6	0
BPOS (WORD)	5.5	1.6	0.70	5.5	1.9	0.7	2.4	0.8	0.3	2.4	0.7	0.30
BPOS (DWORD)	6.3	1.6	1.5	6.2	1.9	1.54	2.7	0.8	0.66	2.7	0.7	0.66
数据移动												
MOVE (BIT)	4.7	1.4	0.03	4.2	1.7	0.02	1.8	0.7	0.01	1.7	0.6	0
MOVE (WORD)	3.7	1.4	0.02	3.6	1.7	0.02	1.5	0.7	0.01	1.6	0.6	0.01
MOVE (DWORD)	3.7	1.4	0.04	3.8	1.6	0.04	1.6	0.7	0.02	1.6	0.6	0.02
MOVE_REAL	3.7	1.4	0.05	3.7	1.7	0.04	1.6	0.7	0.02	1.6	0.6	0.02
BLKMOV (WORD)	5.4	2.3	0	5.4	2.6	0	2.3	1.1	0	2.3	1.0	0
BLKMOV (DWORD)	5.7	2.3	0	5.8	1.6	0	2.5	1.1	0	2.4	1.0	0
BLKMOV (UINT)	5.3	2.3	0	5.3	2.6	0	2.3	1.1	0	2.3	1.0	0
SWAP (WORD)	4.0	1.4	0.13	3.7	1.7	0.13	1.6	0.7	0.05	1.6	0.6	0.05
SWAP (DWORD)	4.0	1.4	0.17	3.8	1.6	0.17	1.7	0.7	0.07	1.7	0.6	0.07

	CPU310			CPE010			CPE020			CRE020		
Function	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***
BLKCLR	2.9	1.1	0.06	2.9	1.4	0.06	1.2	0.6	0.02	1.2	0.5	0.02
BITSEQ	5.9	5.9	0	5.4	5.4	0	2.3	2.3	0	2.4	2.4	0
SHFR (BIT)	8.7	4.8	0.06	8.6	4.9	0.06	3.7	2.1	0.02	3.9	2.2	0.02
SHFR (WORD)	9.5	6.8	0.20	9.7	7.0	0.20	4.2	3.02	0.08	3.7	2.6	0.09
SHFR (DWORD)	9.5	6.9	0.23	10	7.0	0.24	4.3	3.0	0.1	3.8	2.6	0.10
数据表												
SORT (INT)	46.5	1.4	2.8	46.0	1.7	2.80	19.7	0.7	0.72	20.8	0.6	1.2
SORT (UINT)	46.5	1.5	2.8	46.2	1.7	2.81	19.8	0.7	0.72	20.8	0.6	1.2
SORT (WORD)	46.3	1.5	2.8	46.0	1.7	2.81	19.7	0.7	0.72	20.8	0.7	1.2
TBLRD (INT)	6.2	1.9	0	5.9	2.2	0	2.5	1.0	0	2.5	0.9	0
TBLRD (DINT)	6.2	1.9	0	5.9	2.3	0	2.5	1.0	0	2.5	0.9	0
TBLWRT (INT)	6.1	1.8	0	6.2	2.0	0	2.7	0.84	0	2.7	0.8	0
TBLWRT (DINT)	6.3	1.7	0	6.4	2.0	0	2.7	0.8	0	2.74	0.8	0
FIFORD (INT)	6.1	1.9	0.02	6.2	1.9	0.02	2.6	0.7	0.01	2.7	0.8	0.01
FIFORD (DINT)	6.0	1.9	0.04	6.2	1.9	0.04	2.6	0.8	0.02	2.6	0.8	0.02
FIFOWRT (INT)	4.7	1.4	0	5.2	1.6	0	2.2	0.7	0	2.2	0.6	0
FIFOWRT (DINT	5.0	1.3	0.01	5.2	1.6	0	2.2	0.7	0	2.2	0.6	0
LIFORD (INT)	5.8	1.9	0	5.8	1.9	0	2.4	0.8	0	2.5	0.8	0
LIFORD (DINT)	5.8	1.9	0	5.8	1.9	0	2.5	0.8	0	2.5	0.8	0
LIFOWRT (INT)	4.8	1.4	0	5.2	1.6	0	2.2	0.7	0	2.2	0.6	0
LIFOWRT (DINT)	4.8	1.3	0	5.2	1.6	0	2.2	0.7	0	2.2	0.6	0
LIFOWRT (DWORD)	4.8	1.4	0	5.2	1.6	0	2.6	0.7	0	2.2	0.6	0
数组												
ARRAY_MOVE (BIT)	6.7	2.2	0.02	6.5	2.5	0.02	2.8	1.1	0.01	2.8	1.0	0.01
ARRAY_MOVE (BYTE)	6.1	2.2	0.01	5.9	2.4	0.01	2.6	1.0	0	2.6	1.0	0
ARRAY_MOVE (WORD)	6.0	2.2	0.020	5.9	2.4	0.02	2.5	1.0	0.01	2.6	1.0	0.01
ARRAY_MOVE (DWORD)	6.0	2.2	0.04	5.9	2.4	0.04	2.5	1.0	0.02	2.5	0.9	0.02
SRCH (BYTE)	6.3	2.0	0.06	6.8	2.3	0.05	2.9	1.0	0.02	3.0	0.9	0.02
SRCH (WORD)	6.4	2.0	0.07	6.8	2.3	0.07	2.9	1.0	0.03	2.7	0.9	0.03
SRCH (DWORD)	6.5	2.0	0.06	6.3	2.3	0.07	2.7	1.0	0.03	2.8	0.9	0.02
ARRAY_RANGE (WORD)	6.6	2.1	0.61	6.2	2.3	0.61	2.7	1.0	0.26	2.7	0.9	0.26
ARRAY_RANGE (DWORD)	6.7	2.0	0.65	6.4	2.2	0.65	1.5	1.0	0.28	2.8	0.9	0.27
转换												
to INT (UINT)	2.9	1.1	0	2.8	1.4	0	1.2	0.6	0	1.2	0.5	0
to INT (DINT)	2.9	1.1	0	2.8	1.4	0	1.2	0.6	0	1.3	0.5	0
to INT (BCD-4)	3.1	1.1	0	3.1	1.4	0	1.3	0.6	0	1.5	0.7	0
to DINT (INT)	3.0	1.2	0	2.8	1.5	0	1.2	0.6	0	1.5	0.7	0
to DINT (UINT)	2.9	1.1	0	2.8	1.4	0	1.2	0.6	0	1.3	0.5	0
to DINT (BCD-8)	3.7	1.1	0	3.6	1.4	0	1.5	0.6	0	1.5	0.5	0
to UINT (INT)	2.9	1.2	0	2.8	1.4	0	1.2	0.6	0	1.3	0.5	0
to UINT (DINT)	2.9	1.1	0	2.9	1.4	0	1.2	0.6	0	1.3	0.5	0
to UINT (BCD-4)	3.2	1.1	0	3.1	1.4	0	1.3	0.6	0	1.4	0.5	0
to BCD-4 (INT)	3.2	1.1	0	3.1	1.5	0	1.3	0.6	0	1.4	0.5	0
to BCD-4 (UINT)	3.3	1.2	0	3.3	1.4	0	1.4	0.6	0	1.5	0.5	0
to BCD-8 (DINT)	3.3	1.1	0	3.4	1.4	0	1.5	0.6	0	1.4	0.5	0
REAL_TO_INT	3.7	1.1	0	3.8	1.4	0	1.6	0.6	0	1.5	0.5	0
REAL_TO_UINT	3.6	1.1	0	3.5	1.4	0	1.5	0.6	0	1.6	0.5	0
REAL_TO_DINT	3.6	1.1	0	3.7	1.4	0	1.6	0.6	0	1.6	0.5	0
INT_TO_REAL	3.0	1.1	0	3.0	1.4	0	1.3	0.6	0	1.3	0.5	0
UINT_TO_REAL	3.1	1.2	0	3.3	1.6	0	1.3	0.6	0	1.3	0.5	0
DINT_TO_REAL	3.0	1.1	0	3.2	1.4	0	1.3	0.6	0	1.3	0.5	0

	CPU310			CPE010			CPE020			CRE020		
Function	**Enabled**	**Disabled**	**Increment**	**Enabled**	**Disabled**	**Increment**	**Enabled**	**Disabled**	**Increment**	**Enabled**	**Disabled**	**Increment**
REAL_TRUN_INT	-	-	-	-	-	-	-	-	-	-	-	-
REAL_TRUN_DINT	-	-	-	-	-	-	-	-	-	-	-	-
DEG_TO_RAD	3.0	1.1	0	3.0	1.4	0	1.3	0.6	0	1.2	0.5	0
RAD_TO_DEG	2.9	1.1	0	2.9	1.4	0	1.3	0.6	0	1.2	0.5	0
BCD4_TO_REAL	3.2	1.1	0	3.2	1.4	0	1.4	0.6	0	1.4	0.5	0
BCD8_TO_REAL	3.7	1.5	0	3.6	1.4	0	1.5	0.6	0	1.6	0.5	0
控制												
JUMPN	0.3	0.3	0	0.2	0.3	0	0.1	0.1	0	0.1	0.1	0
FOR/NEXT	1.6	0.7	0	1.5	0.7	0	0.7	0.31	0	0.6	0.4	0
MCRN/ENDMCRN Combined	0.8	0.5	0	3.3	0.6	0	0.3	0.3	0	0.3	0.3	0
DOIO	127.1	1.4	0	88.1	1.6	0	49.4	0.7	0	51.2	0.6	0
DOIO with ALT	126.7	1.4	0	87.7	1.5	0	49.2	0.7	0	51.1	0.6	0
SUSIO	3.0	0.8	0	0.8	0.8	0	1.4	0.3	0	1.1	0.3	0
COMMREQ	11.4	1.6	0	8.4	1.8	0	3.6	0.8	0	4.1	0.8	0
CALL/RETURN (LD)	11.0	0.5	0	11.7	0.5	0	5.0	0.2	0	5.0	0.3	0
CALL/RETURN (PSB)	6.8	0.5	0	7.8	0.5	0	3.3	0.2	0	3.2	0.2	0
CALL/RETURN (C Block)	11.9	0.60	0	11.7	0.6	0	5.0	0.3	0	5.1	0.3	0
PIDISA	9.1	8.4	0	9.5	8.5	0	4.1	3.6	0	4.1	3.7	0
PIDIND	9.1	8.3	0	9.5	8.5	0	4.1	3.6	0	4.0	3.7	0
BUS_RD (BYTE)*	NA	NA	NA	22.2	1.3	0	10.9	0.6	0	10.9	0.6	0
BUS_RD (DWORD)*	NA	NA	NA	22.2	1.2	0	11.0	0.5	0	11.0	0.5	0
BUS_RD (WORD)*	NA	NA	NA	22.2	1.2	0	10.9	0.5	0	10.9	0.5	0
BUS_WRT (BYTE)*	NA	NA	NA	22.8	1.3	0	11.4	0.5	0	11.4	0.5	0
BUS_WRT (DWORD)*	NA	NA	NA	23.0	1.3	0	11.4	0.5	0	11.4	0.5	0
BUS_WRT (WORD)*	NA	NA	NA	22.9	1.3	0	11.4	0.5	0	11.4	0.5	0
BUS_RMW (BYTE)*	NA	NA	NA	23.6	1.0	0	12.1	0.4	0	12.1	0.4	0
BUS_RMW (DWORD)*	NA	NA	NA	23.9	1.1	0	12.3	0.4	0	12.3	0.4	0
BUS_RMW (WORD)*	NA	NA	NA	23.9	1.0	0	12.2	0.4	0	12.2	0.4	0
BUS_TS (BYTE)*	NA	NA	NA	23.4	0.2	0	12.0	0.1	0	12.0	0.1	0
BUS_TS (WORD)*	NA	NA	NA	23.7	0.2	0	12.2	0.1	0	12.2	0.1	0

* Results will vary with how quickly the module responds to VME cycles. These times were measured using a Series 90-70 Genius Bus Controller.

	CPU310			CPE010			CPE020			CRE020		
Function	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***	***Enabled***	***Disabled***	***Increment***
SVCREQ												
#1 更改/读取定常扫描定时器	31.6	1.1	0	40.0	1.3	0	14.5	0.6	0	31.9	0.5	0
#2 读取窗口模式和时间	41.3	1.1	0	44.4	1.4	0	19.0	0.6	0	44.5	0.5	0
#3 更改通讯窗口模式和定时器值	18.9	1.1	0	21.0	1.4	0	9.0	0.6	0	17.8	0.6	0
#4 更改背板通讯窗口模式和定时器值	19.3	1.2	0	21.0	1.5	0	9.0	0.7	0	17.9	0.6	0
#5 更改后台任务窗口模式和定时器值	19.2	1.1	0	21.2	1.4	0	9.1	0.6	0	18.0	0.6	0
#6 更改/读取检查字数	19.2	1.3	0	21.0	1.5	0	9.0	0.6	0	17.8	0.6	0
#7 更改或读取当前时间	13.4	1.2	0	11.9	1.7	0	5.1	0.7	0	4.9	0.7	0
#8 重启看门狗定时器	11.5	1.1	0	11.8	1.5	0	5.9	0.6	0	5.6	0.5	0
#9 从扫描开始时读取扫描时间	9.5	1.1	0	8.5	1.5	0	3.6	0.6	0	3.5	0.5	0
#10 读取项目名称	8.6	1.2	0	7.5	1.5	0	3.2	0.6	0	3.2	0.5	0
#11 读取 PLC ID	6.0	1.1	0	5.2	1.4	0	2.2	0.6	0	2.3	0.5	0
#12 读取 PLC 运行状态	4.0	1.2	0	3.7	1.4	0	1.6	0.6	0	1.6	0.5	0
#13 停掉 PLC	17.2	1.1	0	19.1	1.4	0	8.2	0.6	0	17.0	0.6	0
#14 清除 PLC 或 I/O 故障表	519.3	1.1	0	511.5	1.4	0	227.5	0.6	0	220.3	0.6	0
#15 读取最近一次的故障	4.9	1.2	0	19.1	1.3	0	2.1	0.6	0	9.0	0.5	0
#16 读取逝去时间时钟	8.2	1.2	0	7.8	1.5	0	3.3	0.7	0	3.1	0.5	0
#17 Mask/Unmask I/O 中断	5.8	1.2	0	5.3	1.4	0	2.3	0.6	0	NA	NA	NA
#18 读 I/O Override 状态	254.4	1.2	0	253.6	1.5	0	108.6	0.6	0	108.2	0.5	0
#19 设定 Run Enable/Disable	17.3	1.2	0	18.7	1.5	0	8.0	0.6	0	17.1	0.5	0
#20 读取故障表	25.9	1.2	0	28.1	1.5	0	11.1	0.6	0	12.2	0.5	0
#21 用户定义故障记录	58.5	1.2	0	56.0	1.4	0	23.9	0.6	0	33.5	0.5	0
#22 Mask/Unmask 定时中断	4.5	1.2	0	4.1	1.4	0	1.8	0.6	0	1.9	0.5	0
#23 读取主控器 检测	127.0	1.2	0	127.5	1.4	0	54.5	0.6	0	54.5	0.5	0
#25 Disable/Enable EXE 块和独立 C 程序检测	6.1	1.1	0	6.4	1.4	0	2.8	0.6	0	2.4	0.5	0
#32 暂停/继续 I/O 中断	-	-	-	6.2	1.4	0	2.6	0.6	0	NA	NA	NA
#50 读逝去时间时钟	8.2	1.1	0	7.7	1.4	0	3.3	0.6	0	3.1	0.5	0
#51 从扫描开始时读取扫描时间	9.1	1.1	0	8.9	1.4	0	3.8	0.6	0	3.5	0.5	0

总计扫描影响时间

本部分包含 RX7i CPU 的时间信息。将这些信息综合起来即可预测 CPU 的扫描时间。本部分的信息由每种 CPU 模式的基本扫描时间和影响时间组成。预测的扫描时间为基本执行时间、扫描影响时间和预估的逻辑执行时间的总和。

A-22 页列举了 RX7i 扫描时间预估的例子。

扫描影响时间有以下五个部分组成：

- 编程器通讯扫描影响
- I/O 扫描和故障扫描影响
- 以太网全局数据扫描影响
- 智能选择模块(LAN 模块)扫描影响
- I/O 中断性能和扫描影响

基本扫描时间

基本扫描时间是空_MAIN 块，无配置，无窗口激活状况下的执行时间。下表是每种 CPU 样式的基本扫描时间。I/O enabled 情况下基本扫描时间包括扫描以太网子板的时间。

基本扫描时间

CPU 样式	CPE020	CPE010
基本扫描时间 Stop + I/O enabled	220 µsecs	465 µsecs
基本扫描时间 Stop + I/O disabled	186 µsecs	402 µsecs

下表描述了全过程扫描和基本扫描的区别：

基本扫描 vs. 全过程扫描

基本扫描	全过程扫描
<开始扫描>	<开始扫描>
扫描内部处理	扫描内部处理
↓	↓
空的输入扫描 *	输入扫描 *
↓	↓
执行程序逻辑	EGD 接收扫描***
↓	↓
空的输出扫描*	执行程序逻辑
↓	↓
↓	输出扫描 *
↓	↓
↓	EGD 发出扫描 ***
↓	↓
↓	验证失去的 I/O 模块 **
↓	↓
↓	控制器通讯窗口
↓	↓
<扫描结束>	背板通讯窗口
	<扫描结束>

* 如果 I/O 暂停，则跳过输入输出扫描
** 只在发生丢失某模块故障时才检测丢失的 I/O 模块
*** 如果没有配置以太网全局数据交换，跳过发送和接收扫描。

对于基本扫描，如果没有配置，扫描的输入输出扫描阶段为空(例如，检测配置之后结束)。配置中无 I/O 模块(GBC)和智能模块时效果相同。基本扫描的逻辑执行时间不为 0。将 _MAIN 块的时间计入基本扫描时间，用户只需将实际程序的执行时间加上即可。基本扫描时间为假设没有丢失 I/O 模块的情况下得到的。没有编程器连接意味着控制器通讯窗口没有打开。没有只能选择模块意味着背板通讯窗口没有打开。

表的内容

在一些表中，功能设为不同步影响扫描。这意味着这些功能不是扫描的一个阶段。例如，I/O 模块扫描发生在 CPU 扫描的输入或者输出阶段。然而，I/O 中断通常与扫描不同步并且会中断当前进程的任何功能。

通讯功能(除了高级别的编程器请求)都在扫描过程中两个窗口(控制器通讯窗口和背板通讯窗口)中的中的一个窗口内执行。所有的服务请求的扫描影响时间都是定义的时间里最小的，窗口时间会被调节以便没有哪个窗口的时间片(限制的)会在特定的扫描内发生。这意味着，尽最大可能将每个功能在窗口的一次执行过程中完成(在连续的逻辑扫描过程中)。扫描影响时间可以在多个扫描过程中完成(受限制的)，只要将窗口时间调整到小于扫描影响时间即可。对于编程器，缺省时间为 10 毫秒，因此，那个部分所列的部分功能自然是要在多个连续的扫描过程中完成。

编程器扫描影响时间

下表列出了编程器名义扫描影响时间，单位为毫秒。

编程器扫描影响时间

扫描影响条目	描述	CPE020	CPE010
编程器窗口	打开编程器窗口但不处理任何请求的时间。编程器通过以太网联接，不监控变量值。	0.24 µsec	1.46 µsec
变量表监控器	更新变量表屏幕的扫描影响 (%R 变量表为例)。混合表显示影响稍大。扫描影响可能不连续，依赖于 CPU 扫描时间和编程软件主机的执行速度。	0.37 µsec	2.10 µsec
编辑器监控器	监控梯形图逻辑时更新编程器屏幕的扫描影响。表中列的是 1 个结点，2 个线圈和 11 个寄存器的显示更新时间。和变量表扫描影响一样，这个影响可能也是不连续的。	0.30 µsec	1.49 µsec

I/O 扫描和 I/O 故障扫描影响

I/O 扫描影响有 2 个部分，本地 I/O 和 Genius I/O。计算 I/O 扫描的影响的等式为：

I/O 扫描影响 = 本地扫描影响 + Genius I/O 扫描影响

本地 I/O 模块的扫描影响

模块位置和变量地址和模块的数量同样影响 I/O 模块的 I/O 扫描。I/O 扫描有几个基本的部分：

I/O 扫描	***描述***
机架设定时间	因为扩展机架地址在 VME 总线上，所以每一个扩展机架单独选择。不管机架上模块的具体数目，每个扩展机架有一个固定的总计时间。
每个模块的设定时间	每一个本地 I/O 模块有一个固定的设定扫描时间。
字节传输时间	主机架内模块的字节的实际传输速度比扩展机架模块实际字节传输速度要快得多。模块的字节传输速度根据机架的不同而确定。

只要具有连续槽位地址的模块使用连续的变量地址，模拟量输入放大模块(如同 Genius 模块一样)可以编组传输数据。每组连续地址模块称作一个扫描段。每增加一个扫描段都会额外消耗扫描时间。

RX7i I/O 模块类型

类型	*零件编号*
离散输入类型 I (16 点, 14 点)	IC697MDL240, IC697MDL241, IC697MDL251, IC697MDL640, IC697MDL671
离散输入类型 II (32 点)	IC697MDL250, IC697MDL252, IC697MDL253, IC697MDL254, IC697MDL651, IC697MDL652, IC697MDL653, IC697MDL654
离散输出类型 I (16 点, 12 点)	IC697MDL340, IC697MDL341, IC697MDL740, IC697MDL940
离散输出类型 II (32 点)	IC697MDL350, IC697MDL750, IC697MDL752, IC697MDL753
模拟量输入类型 I (8 通道)	IC697ALG230
模拟量输入类型 II (带 8 通道 AI 的 16 通道模块)	IC697ALG440, IC697ALG441
Analog Output 模拟量输出 (4 通道)	IC697ALG320

RX7i 模块扫描影响时间

注意: 基本情况提供了机架上只有一个模块式的总计时间。increment (Inc)值参考增加的每个小模块的总计时间。

扫描影响时间（毫秒）								
	CPE020				***CPE010***			
	主机架		*扩展机架*		*主机架*		*扩展机架*	
	主机架	*主机架(Inc)*	*扩展机架*	*扩展机架 (Inc)*	*主机架*	*主机架(Inc)*	*扩展机架*	*扩展机架 (Inc)*
离散输入 类型 - I	18.803	16.716	19.639	22.922	37.244	29.224	41.782	46.219
离散输入 类型- II	20.554	15.559	23.031	27.854	37.899	30.292	48.323	48.175
离散输出 类型- I	20.419	16.275	20.025	22.775	38.432	30.664	45.227	45.361
离散输出 类型- II	21.039	17.653	22.938	29.411	40.232	32.464	48.288	49.274
每个故障信息	44.608		45.716		106.254		111.01	
模拟量输入– 类型 1	25.285	24.004	45.04	48.319	49.28	45.396	82.44	66.533
模拟量扩展 – 类型 2	37.712	13.797	103.843	55.339	64.709	14.741	137.88	62.641
模拟量输出	24.723	20.693	34.217	37.704	49.723	43.004	70.976	55.912
每个故障信息	40.135		60.762		86.207		86.7	

工作页 A: I/O 模块扫描时间

下列表格可以用于 I/O 模块扫描影响时间计算。计算包含模拟量输入放大单元的时间，放大单元可以是单个计算的，也可以是编组计算的。扫描影响时间可以在 A-错误！书签未定义"页，错误！没找到变量来源"。

扩展机架号	______	
每个扩展机架的扫描影响	x ______	= ______
主机架离散 I/O 模块数	______	
主机架每个离散 I/O 模块扫描影响	x ______	= ______
扩展机架离散 I/O 模块数	______	
扩展机架每个离散 I/O 模块扫描影响	x ______	= ______
主机架模拟 I/O 模块数	______	
主机架每个模拟 I/O 模块扫描影响时间	x ______	= ______
主机架模拟量输入放大模块数(同一段)	______	
主机架每个模拟量输入放大模块 (同一段) 扫描影响时间	x ______	= ______
主机架模拟量输入放大模块数(新的段)	______	
主机架每个模拟量输入放大模块 (新的段) 扫描影响时间	x ______	= ______
扩展机架模拟 I/O 模块数	______	
扩展机架每个模拟 I/O 模块扫描影响	x ______	= ______
扩展机架模拟量输入放大模块数(同一段)	______	
扩展机架每个模拟量输入放大模块 (同一段) 扫描影响时间	x ______	= ______
扩展机架模拟量输入放大模块数(新的段)	______	
扩展机架每个模拟量输入放大模块 (新的段) 扫描影响时间	x ______	= ______
预计的 I/O 模块扫描影响时间		______

注意: 如果点故障使能，则下列与点故障对应的点故障时间使能，如下表所示。

下表列出了每点或每个通道大概所需时间。这些平均值基于 1024 个离散点(512 输入，512 输出)和 128 通道模拟量(96 输入，32 输出)。实际的值会和平均值有所不同，这依赖于系统 I/O 配置。

Genius I/O 和 GBC 的扫描影响

对于 Genius I/O 和 Genius 总线控制器(GBC)，每一个 GBC 和每一个扫描段都有扫描影响时间，并且所有的 I/O 数据传递时间都对扫描时间造成影响。

扫描影响有 3 个部分：

1. 打开系统通讯窗口的扫描影响。只在第 1 个智能选择模块(GBC 是 1 个)加入系统时计入。
2. 用后台信息(数据表)校验每个 GBC 的扫描影响。这部分是系统内每个 GBC 的影响。

注意: 关闭后台通讯窗口(将其时间设为 0)可以消除以上两种 GBC 扫描影响。这种方式只能用于对时间要求非常严格，要求确保最小扫描时间的过程。窗口关闭时，收到的信息会超时，COMM_REQ 停止工作。

3. 扫描的 GBC 的影响。这种影响来源于 CPU 通知 GBC 自己的新的输出数据已经传输并且命令 GBC 准备好自己的输入数据，同样也通知 GBC 自己已经完成另一个扫描并且仍然在运行模式。

Genius I/O 的一个扫描段由同一总线上总线地址连续，变量地址也连续的 Genius 模块组成。单个扫描段的输入扫描时间长于输出扫描时间。模拟量，离散量和全局数据扫描段的扫描进程是相同的。离散数据以字节为单位传输，比模拟量消耗的时间要长，模拟量是以字为单位进行传输的。全局数据可以以离散量来计算，也可以以模拟量来计算，具体要看数据源头和目的地所使用数据的变量类型。

Genius I/O 和 GBC 扫描影响时间

注意: 粗体字显示的功能持续影响扫描。其他功能只在激活时影响扫描。不是所有下表列举的定时信息在手册印刷时仍然可用(空白处)

	CPE020 (µsec)	***CPE010 (µsec)***
Genius 总线控制器		
打开背板通讯窗口	--	--
每一个 Genius 总线控制器校验后台消息	--	--
每一个 Genius 总线控制器 I/O 扫描	--	--
第一个 Genius 总线控制器**	--	--
后面的 Genius 总线控制器	--	--
Genius I/O 模块		
每一个 I/O 模块扫描段	172	180
每一个 I/O 模块扫描段字/点故障使能	178	185
主机架每个字节离散 I/O 数据	3	3
扩展机架每个字节离散 I/O 数据	4	4
主机架每个字模拟 I/O 数据	20	30
扩展机架每个字模拟 I/O 数据	20	30

可以用下列表格预测 Genius I/O 扫描影响时间。在 A-15 页“Genius I/O 和 GBC 扫描影响时间”处可以找到扫描影响时间部分内容。

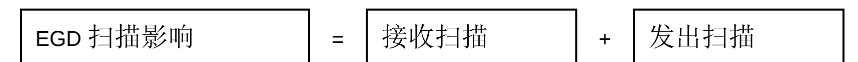
工作表 B: Genius I/O 扫描影响时间

打开背板通讯窗口		______	=	______
GBC I/O 扫描		______		
GBC 校验后台信息	+	______	=	______
GBC 的数量	x	______	=	______
输入模块扫描段数		______		
输入模块扫描段扫描影响	x	______	=	______
输出模块扫描段数		______		
输出模块扫描段扫描影响	x	______	=	______
主机架上 GBC 的离散 I/O 数据字节数		______		
主机架上的离散 I/O 每个字节的扫描影响	x	______	=	______
扩展机架上 GBC 的离散 I/O 数据字节数		______		
扩展机架上的离散 I/O 每个字节的扫描影响	x	______	=	______
主机架上 GBC 的模拟 I/O 数据字数		______		
主机架上的模拟 I/O 每个字的扫描影响	x	______	=	______
扩展机架上 GBC 的模拟 I/O 数据字数		______		
扩展机架上的模拟 I/O 每个字的扫描影响	x	______	=	______
预测 Genius I/O 扫描影响				______

以太网全局数据扫描影响

依赖于 CPU 扫描时间和以太网全局数据(EGD)交换周期,以太网全局数据可能每个周期传输，也可能隔几个周期传输一次。因此，扫描影响会因扫描时的交换数不同而不同。在计算最坏情况下的扫描影响时，必须将所有的交换计入在内。

以太网全局数据(EGD)扫描影响有 2 个部分，接收扫描和发出扫描：



在设定 CPU 定常扫描模式和设定 CPU 看门狗时间时，应将扫描影响计入。

接收和发出扫描有 2 个部分组成，交换时间和字节传输时间：



交换时间

交换时间包括在扫描过程中进行交换的设置时间。当计算扫描影响时，每个交换都要包括这个时间。

注意: 下表所列的总计交换时间是假定的最坏情况下 100 个变量，1400 字节的传输时间。

EGD 总计交换时间		
	内置 以太网接口	***基于机架的 以太网模块***
CPE010接收 /读取	135.7 µsec	184.4 µsec
发出 / 写入	182.1 µsec	255.8 µsec
CPE020接收 /读取	64.1 µsec	99.2 µsec
发出 / 写入	99.7 µsec	143.1 µsec

数据传输时间

CPU 模块和以太网模块之间传输数据所需的时间。

注意: EGD 数据传输时间和数据量的大小之间不是线性关系。清是用下表所述数据值来估计传输时间。

数据传输时间				
CPU	*数据量 (字节)*	*方向*	*内置以太网接口*	*基于机架的以太网模块*
CPE010	1	接收/ 读取	2.5 µsec	4.4 µsec
	100	接收/ 读取	25.5 µsec	17.6 µsec
	200	接收/ 读取	48.9 µsec	32.2 µsec
	256	接收/ 读取	62.2 µsec	39.5 µsec
	1	发送 /写入	1.5 µsec	4.3 µsec
	100	发送 /写入	3.3 µsec	11.1 µsec
	200	发送 /写入	5.3 µsec	16.3 µsec
	256	发送 /写入	6.5 µsec	18.9 µsec
CPE020	1	接收/ 读取	1.6 µsec	3.3 µsec
	100	接收/ 读取	23.7 µsec	16.7 µsec
	200	接收/ 读取	46.2 µsec	31.4 µsec
	256	接收/ 读取	59.1 µsec	38.9 µsec
	1	发送 /写入	0.7 µsec	3.6 µsec
	100	发送 /写入	2.5 µsec	10.7 µsec
	200	发送 /写入	4.5 µsec	15.8 µsec
	256	发送 /写入	5.6 µsec	18.3 µsec

工作表C:以太网全局数据扫描时间

接收的交换数		______		
每个交换的扫描影响	x	______	=	______
所有接收的交换数据的字节数		______		
接收的每个字节的扫描影响	x	______	=	______
发送的交换数		______		
每个交换的扫描影响	x	______	=	______
所有发送的交换数据的字节数		______		
发送的每个字节的扫描影响	x	______	=	______
		预测的以太网全局数据扫描影响		______

智能选择模块扫描影响

智能选择模块包括用于 Genius LAN 的 GBC 模块。这些智能选择模块的扫描影响变化范围很大。打开背板通讯窗口和校验每个模块的扫描影响相对于 CPU 存储器读写请求来讲非常小。

下面这个等式说明了如何计算每个模块的固定扫描。

GBC	=	依次检测扫描影响 + GBC I/O 扫描影响

下表描述了智能选择模块的固定扫描影响时间(毫秒)

智能选择模块扫描影响时间

扫描影响条目	CPE020	CPE010
ETM (外围以太网模块)	30 µsec	55 µsec
告诉计数器	150.9679 µsec	252.4117 µsec
GBC	见 A-15页"Genius I/O 和 GBC 扫描影响时间 的扫描影响时间"	见 A-15页"Genius I/O 和 GBC 扫描影响时间 的扫描影响时间

I/O 中断性能和扫描影响

I/O 中断模块有几个重要的性能号码。激活一个空的程序块的 I/O 中断的扫描影响测量的是守备中断时间，启动模块时间，退出模块时间和重启中断任务的时间总和。执行中断块中的逻辑所需的时间会使 CPU 花更多的时间来进行 I/O 中断服务，这样会减小最大中断速率。

最小，典型和最大的中断响应时间反映了从单个 I/O 模块接收到输入脉冲到中断块中第一行逻辑执行时刻的时间间隔。最小响应时间反映了 300 毫秒输入卡过滤器时间+中断发生到中断块的第一行逻辑执行的时间间隔。只有系统内没有智能选择模块并且编程器没有连接时才能达到最小时间响应。典型响应时间为最小响应时间+2 毫秒的最大的中断反应时间。中断反应时间是正确的，除非发生以下情况：

- 与编程器连接
- 存储逻辑，运行模式存储或者进行逐字替换时
- 发生故障(登入故障)时
- 发生另一个 I/O 中断时
- CPU 从 I/O 控制器(比如 GBC)传输很大数量的输入(或输出)数据时。可能的话，尽量把负载比较大的 I/O 控制器放在主机架。

以上任何一个事件发生都会增大中断反应时间(从中断卡发出信号给 CPU 到 CPU 进行中断服务的时间)，使中断反应时间超出典型值。然而，由于前一个中断执行时间不确定，所以中断反应时间不固定。I/O 中断顺序执行，所以单个 I/O 中断的反应时间受前面的中断块执行时间影响。(最坏的情况是系统中所有的 I/O 中断同时发生，则最后一个执行的中断必须等待其他中断执行完成才能执行)

下列的最大响应时间不包括这两种极大事件

I/O 中断块性能和扫描影响时间

扫描影响条目	**CPE020 (µsec)**	**CPE010 (µsec)**
中断扫描影响	198.0	364.8
最小响应时间 典型响应时间 最大响应时间	367.8 375.5 410.5	449.5 463.7 510.7

注意最小，典型和最大响应时间包括 300 微秒输入卡过滤时间.

下列工作表用于计算编程器，智能选择模块和 I/O 中断扫描影响时间。关于时间数据参考下表：

编程器描影响时间, A-错误！未定义书签。页

RX7i错误！未找到引用源。, A-错误！未定义书签。页

Genius I/O 和 GBC的扫描影响时间, A-15页

工作表 D:编程器, IOM, I/O 中断扫描时间

编程器扫描影响			= ______
IOM—第一个模块(打开通讯窗口)		______	
IOM—每个模块(轮流检测)		+ ______	
LAN 模块 I/O 扫描		+ ______	
	总计 IOM 扫描影响		= ______
从 IOM 进行的 CPU 存储器访问			= ______
I/O 中断扫描影响		______	
I/O 中断响应时间		+ ______	= ______
	预测的扫描时间(其他的)		______

定时中断性能

激活一个空的程序块的 I/O 中断的扫描影响测量的是守备中断时间，启动模块时间，退出模块时间和重启中断任务的时间总和。最小，典型和最大的中断响应时间反映了从单个 I/O 模块接收到输入脉冲到中断块中第一行逻辑执行时刻的时间间隔。只有系统内没有智能选择模块并且编程器没有连接时才能达到最小时间响应。执行中断块中的逻辑所需的时间会使 CPU 花更多的时间来进行 I/O 中断服务，这样会减小最大中断速率。典型响应时间为最小响应时间+最大的中断反应时间。中断反应时间是正确的，除非发生以下情况：

- 与编程器连接
- 存储逻辑，运行模式存储或者进行逐字替换时
- 发生故障(登入故障)时
- 发生另一个 I/O 中断时

. 以上任何一个事件发生都会增大中断反应时间，使中断反应时间超出典型值。然而，由于前一个中断执行时间不确定，或者 I/O 中断为极大的情况，所以中断反应时间不固定。最坏的情况是系统中所有的 I/O 中断同时发生，则最后一个执行的中断必须等待其他中断执行完成才能执行

下列的最大响应时间不包括这两种极大事件

定时中断和扫描影响时间

扫描影响条目	***CPE020 (μsec)***	***CPE010 (μsec)***
定时中断扫描影响	82.8	152.8
最小响应时间 典型响应时间 最大响应时间	49.7 55.7 77.5	100 114.8 161.5

预测扫描时间计算实例

本例中估计的扫描时间不包括逻辑执行时间。计算的时间是点故障 disabled,编程器未连接情况下正常扫描时间。计算的时间从下列表中得到：

基本扫描时间, A-错误！未定义书签。页

RX7i错误！未找到引用源。, A-错误！未定义书签。页

Genius I/O 和 GBC的扫描影响时间, A-15页

下面的例子为如何预测扫描时间：

RX7i 系统配置举例

PS	CPE010	BTM	32 点输入	32 点输入	32 点输出	32 点输出	8 通道模拟量输入	4 通道模拟量输出	ETM
0	1	2	3	4	5	6	7	8	9

主机架

s 扫描计算

预测的扫描 = 基本扫描 + I/O 扫描影响

基本扫描时间		= 465
I/O 扫描影响…		
主机架上 2 个离散输入模块 每个离散 I/O 模块扫描影响	2 x 37.9	= 75.8
主机架上 2 个离散输出模块 每个离散 I/O 模块扫描影响	2 x 80.4	= 80.4
主机架上模拟量输入模块数 主机架上输出模块扫描影响时间	1 x 49.3	= 49.3
主机架上模拟量输出模块数 主机架上输出模块扫描影响时间	1 x 49.7	= 49.7
以太网模块	1 x .55	= 55.0
	预测扫描时间	= 775.2

注意: 时间为毫秒

Appendix B 用户存储器分配

用户存储器大小是用户 PLC 程序可以使用的存储器字节数。

用户存储器大小	字节数
10MB	10,485,760

计算用户存储器的项目

下列项目计算 CPU 存储器大小，并可用于计算应用程序所需存储器。可能另外需要空间来保存高级用户参数，装载(例如压缩文件)，用户堆和公布符号等附加信息。

寄存器大小 (%R)	字节数 = 配置的%R 变量数 × 2
字存储器大小 (%W)	字节数=配置的%W 变量数× 2
模拟量输入 (%AI)	如果使用点故障功能: 字节数= 配置的%AI 变量数 × 3 如果不使用点故障功能: 字节数=配置的%AI 变量数× 2
模拟量输出(%AQ)	如果使用点故障功能: 字节数=配置的%AQ 变量数× 3 如果不使用点故障功能: 字节数=配置的%AQ 变量数× 2
离散点故障	如果使用点故障功能:字节数= 3072
符号变量存储	为离散符号变量预留的位数： 字节数 = ((配置的离散符号变量位数)/(8 位/字节))* 4 **注意:** 位数乘 4 是为跟踪强制和位变量的其他特性。 为非离散符号变量预留的字节数： 字节数 = (配置的符号变量字数) * (2 字节/字) **注意:** 管理已声明的符号变量需要附加的空间。
以太网全局数据 (包含在 HWC 内)	字节数=如果没有配置以太网全局数据页则为 0
I/O 扫描设定文件 (包含在 HWC 内)	基于使用的扫描设定数 **注意:** 如果应用程序每个周期都扫描所有的 I/O（缺省情况），则需要 32 个字节
用户程序	关于用户程序的细节，见 B-2 页“用户程序存储器使用”

用户程序存储器使用

用户逻辑所需的存储空间包括以下几项。

%L 和%P 程序存储

%L 和%P 占用多大地址要看程序中的使用情况。每个 LD 程序块中%L 或者 %P 变量最大为 8192 个。

依据下列规则，允许在运行模式存储时使用额外的%L 和%P 变量表存储空间。

- 如果本个块中没有使用%L 变量，则%L 存储器大小为 0 字节。如果本个块中使用了%L 变量，则保留的缓存应大于程序中或变量表中使用的最大%L 地址所指示的空间。缺省缓存大小为 256 字节，但可以通过编辑程序块属性的额外本地字 (Extra Local Words)参数来改变缓存大小。
- 同样的规则适用于%P 存储器大小，但是%P 存储器可以在程序的任意程序块内使用。
- 缓存不能使%P 或者%L 变量表超过最大尺寸 8192 字。在那种情况下，使用较小的缓存。
- 如果这些变量的最后存储满足%L/%P 变量表(包括前一次缓存空间)，或者%L/%P 变量表从 0 到非 0 大小条件，你可以增加，改变或删除%L 和/或%P 变量或者运行模式下存储应用程序。
- 停止模式存储时要重新计算%L/%P 变量表的大小

程序逻辑和总计

C(.gefelf)块作为用户程序的一部分计入用户程序的大小。CPU 需要额外的空间来执行 C 块。

程序块基于块的总计加上使用的逻辑和寄存器数据(即, %L).

注意: LD 程序堆栈不计入 CPU 存储器大小。

注意: 如果你的应用程序需要更多的空间执行 LD 逻辑，可以将%P 或者%L 变量改为%R, %W, %AI 或者%AQ 变量。这种更改需要重新编译程序块，并且在停止模式下将程序存储到 CPU.

Appendix C 将 90 系列应用程序转为 PAC 程序

PAC 系统控制器功能特征和 90 系列 PLC 不一致，以太网全局数据(EGD)能力和 90-30 系列 PLC 不一致并作了改善。尽管 PAC 系统不支持 90 系列 PLC 的某些功能，但是 PAC 系统相对 90 系列 PLC 来说功能有了很多的提升。

本章提供以下信息：

关于 CPU 规范，参考第 2 章。关于功能定时信息，参考附录 A。

PAC 系统–90 系列比较

这部分概括了两种控制系统的不同特征和操作。

CPU 操作

种类	90-70 系列	90-30 系列	PAC 系统
CPU 可以在主机架的任何槽内	不支持	不支持	RX3i 支持(需要 2 槽). RX7i 不支持.
逻辑执行	以列为主执行(见下面 "逻辑执行")	以行为主执行	以行为主执行(见下面 "逻辑执行")
清除故障	清除个体故障	清除个体故障	清除 PLC 故障表和 I/O 故障表
IO 模块故障重启	在 HWC 的存储和清除上生成.	不在 HWC 的存储和清除上生成	在 HWC 的存储和清除上生成
%S0020 状态位	不支持	当相关功能使用实型数据成功执行时设为 ON 当两个输入数据中有一个为 NaN (不是数字)时清除	不支持.
访问权限	0~4 级	1~4 级	1~4 级
堆栈溢出	检测到堆栈溢出时进入 Stop/Halt 模式	检测到堆栈溢出时进入 Stop/ Fault 模式	如果剩余的堆栈空间不能支持一个给定的程序块调用，会记录“应用程序堆栈溢出”故障。这种情况下，CPU 不能执行这个块。将这个块的输出设为 FALSE，并且执行这个调用之后的指令。
程序名	没使用	没使用	只读的 LD 程序名，LDPROG1,Plant Edition 软件使用

逻辑

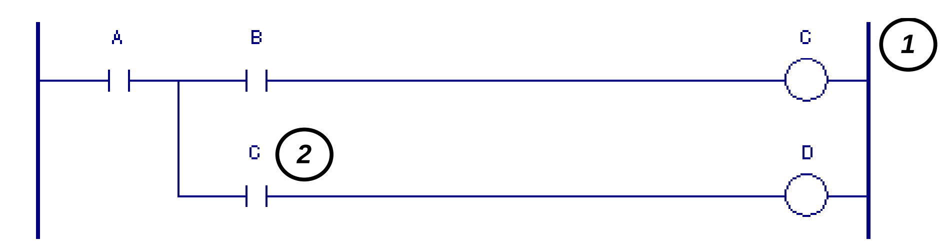
行/列为主的LD 执行

90-70 系列 PLC 以列为主执行：他们执行 LD 逻辑是从上到下，再从左到右执行。90-30，VersaMax 和 PAC 则是先从左到右，再从上到下执行。执行顺序的差异会导致同一回路执行结果不同，这在 90-30 中是不允许的。因此，下列例子不能应用于 90-30。

注意: 90-70 系列项目转化为 PAC 系统项目不会将列为主程序重写为行为主程序。因为行/列为主的不同造成的差异会在转换报告中列出，但是不能保证报告中会列出所有的差异。

例 1

本例中由于执行时行/列为主顺序不同，会造成 C 结点和线圈状态的不同。行为主时，在确定结点(2)的值之前，线圈(1)已经设定了变量 C 的值。列为主时，在确定结点(2)的值之后，才执行线圈(1)。



例 2

本例中，问题没那麽明显。如果变量 C 是 LDBK 的输入输出量，或者是 LDBK 调用的程序块的输入输出量，那麽，行为主还是列为主执行的差异就会影响线圈 C 的执行

行为主时，在执行调用(2)之前，已经执行了 C 线圈(1)。列为主时，在执行调用(2)之后，才执行了 C 线圈(1).

A
B
C
1
CALL LDBK
2
D

例 3

本例中，列为主执行时(90-70 系列)，SUB_INT 总在 ADD_INT 之前执行。行为主执行时(PAC 系统)，SUB_INT 总在 ADD_INT 之后执行。

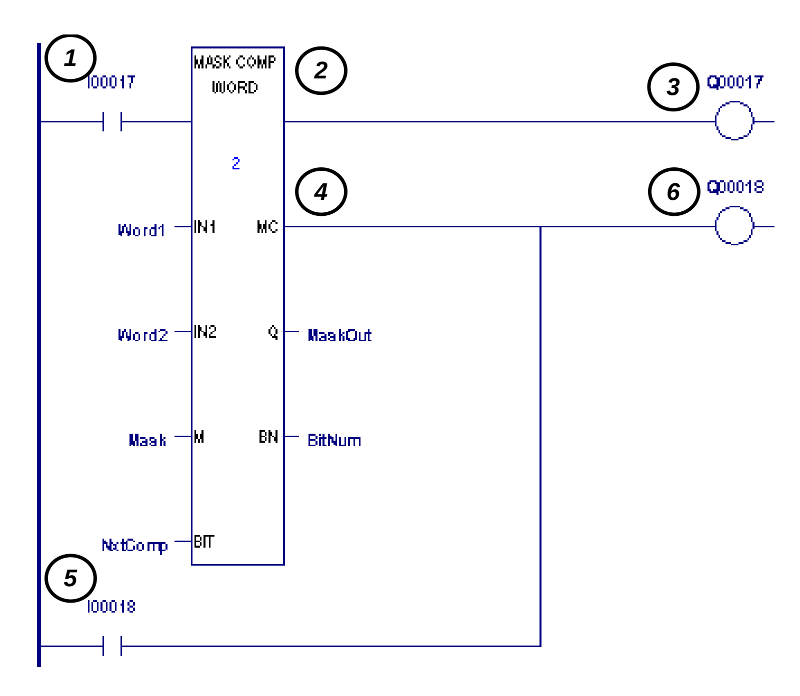
I00001
ADD INT
Q00001
R00001 IN1 Q R00001
1 IN2
SUB INT
R00001 IN1 Q R00001
1 IN2

例 4

本例中，虽然执行顺序不同，但是两种类型的执行结果是一样的。

行为主执行(PAC)

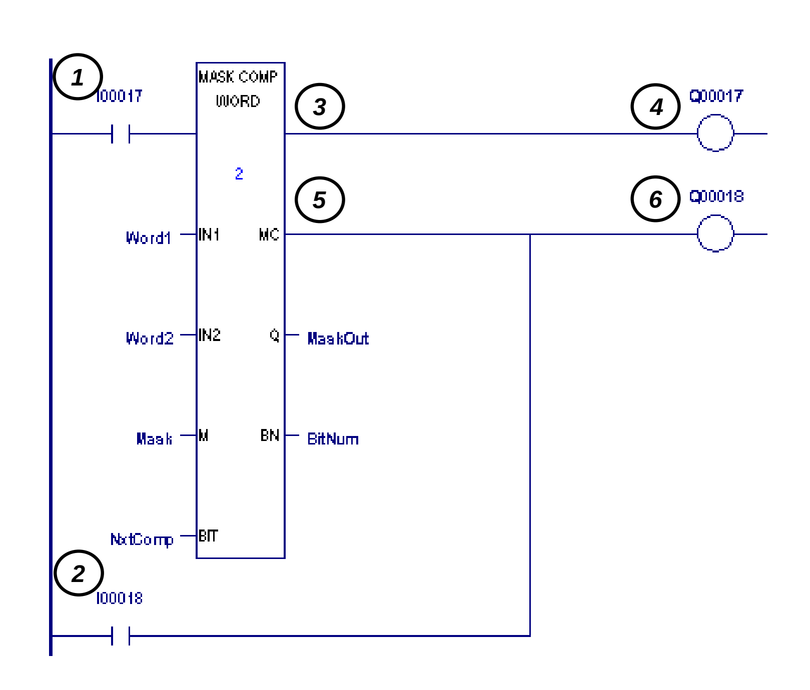
首先确定外部输入量 I00017 (1)的值。如果 I00017 为 1，真伪比较函数（Masked Compare function）(2)执行，传递能流给 Q00017 (3)。如果不同，输出值 MC (4)设为 1。然后确定 I00018 (5)的值。MC 和 I00018 取或之后得到 Q00018 (6)的值。



列为主执行(90-70 系列)

首先确定外部输入量 I00017 (1)的值。再确定外部输入量 I00018(2)的值。如果 I00017 为 1，真伪比较函数（Masked Compare function）(3)执行，传递能流给 Q00017 (4)。如果不同，输出值 MC (5)设为 1。

MC 和 I00018 取或之后得到 Q00018 (6)的值。



程序块最大数目

种类	*90-70 系列*	*90-30 系列*	*PAC 系统*
程序块最大数目	256, 包括_MAIN 块	65, 包括_MAIN 块	512, 包括_MAIN 块

用户程序

种类	*90-70 系列*	*90-30 系列*	*PAC 系统*
C 独立程序	支持	不支持	不支持
LD 程序	支持	支持	支持
结构化文本	不支持	不支持.	对于在特定版本 CPU 是否可用，参考 CPU 附带的 *IPI 文献*
程序编制	支持 5 种模式	只支持顺序(Ordered)模式	只支持顺序(Ordered)模式
中断程序	支持	不支持	不支持
功能块	支持	不支持	对于在特定版本 CPU 是否可用，参考 CPU 附带的 *IPI 文献*
顺序功能表编程	支持	支持	不支持
同步扫描设定	支持	不支持	不支持
FIP,微循环模式和定期程序	支持	不支持	不支持
每个文件夹多个程序	支持(最多 16 个)	不支持	不支持
C 调试器	支持	不支持	不支持
声明逻辑	支持	支持	不支持

堆栈大小

种类	*90-70 系列*	*90-30 系列*	*PAC 系统*
堆栈大小	正确范围: 1~ 64 KB 缺省: 20 KB (LD 程序)	不可配置	正确范围: 8~320 KB, 最小增加值为 8 KB 缺省: 64 KB

C 块

种类	***90-70 系列***	***90-30 系列***	***PAC 系统***
C 开发工具	GFK-0646	GFK-0646	GFK-2259
位	16-位 块	16-位 块	32-位 块
编译好的块的扩展名	.exe or .sta	.exe	.gefelf
最大尺寸	64,000 字节	81,920 字节	只受可用的用户存储器大小限制 **注意:**任何 C 块的大小如果大于 256KB，都会发生确认错误。可以右键单击 C 块，选择属性栏，在 Inspector 对话框中将检查大小限制(Check Size Limits)设定为 FALSE，即可消除这个错误。这样操作有造成存储器段落过小的风险。
倒入	不能倒入为 90-30 和 PAC 开发的程序	不能倒入为 90-70 和 PAC 开发的程序	不能倒入为 90-70 和 90-30 开发的程序。必须将他们重新编辑为扩展名是.gefelf 的文件
名称属性	最多 7 个字符	最多 7 个字符	最多 31 个字符
检查大小限制	不支持	不支持	支持
第三方 VME 中断	支持	不支持	替换为模块中断
模块中断	不支持	不支持	编制模块时从 Triggers 列表中选择。首先在硬件配置中配置模块，然后在模块的参数编辑器的中断键下设定模块的中断 ID 号。
定义的参数	0~7 对输入/输出，输入输出的数目必须相同。	不支持	输入: 0~63. 输出: 1~64. 输入输出数目可以不相同
调用 C 块时可选参数的选择	不支持。你必须为定义的每个参数提供一个值。	不支持	Machine Edition 使你可以允许你讲任何参数空置。但是你有责任确保 C 块使用合适的缺省值并且运行时不发生错误。
字节数据类型	支持	不支持	参考特定 CPU 固件版本的*重要产品信息*
NWORD 数据类型	支持	不支持	不需要。可以用字代替
数据流	不支持	不支持	支持
非直接变量	不支持	不支持	支持
非离散存储器中的位变量	不支持	不支持	支持。必须按字节排列。



块

块的类型可以为普通程序块，参数化块或者功能块

非参数化块

种类	90-70 系列	90-30 系列	PAC 系统
最大尺寸	32 KB	16 KB	128 KB
名称属性	最多 7 个字符	最多 7 个字符	最多 31 个字符
每个程序块分配给%P 或者%L 存储器的额外本地字	不可配置	不支持	可配置
第三方 VME 中断	支持	不支持	替换为模块中断
模块中断	不支持	不支持	编制模块是从列表中选择，首先在硬件配置中配置模块
非离散存储器中的位变量	不支持	不支持	支持

参数化块 (PSBs)

90-30 系列不支持

种类	90-70 系列	PAC 系统
字节数据类型	支持	参考特定 CPU 固件版本的*重要产品信息*
NWORD 数据类型	支持	没有要求. 可以用字替代.
Y0 参数	不能用于 0 参数 PSB。Y0 假定为 TRUE	可以用于任何 PSB 的逻辑，包括一个 0 参数 PSB。可以在程序块中使用，除了_MAIN
布尔参数	不支持长度大于 1 的布尔型输入输出参数。只支持能流，不支持数据流。	数据流和能流都支持
	不支持常数	支持常数作为输入参数
32-位参数 (DINT, DWORD 和 REAL)	不支持常数	支持常数作为输入参数
非离散存储器中的位变量	不支持	支持
模块名称	最多 7 个字符	最多 31 个字符
参数	最多 7 对输入/输出(包括 Y0 的话可以有 8 个输出)	最多 63 个输入和 63 个输出(包括 Y0 的话可以有 64 个输出). 参考特定 CPU 固件版本的*重要产品信息*

功能块 PAC 系统允许使用功能块，功能块由用户定义，具有参数和临时数据。90 系列 PLC 不支持这个特征。

PAC 系统功能

下列介绍的是 PAC 系统的功能，90 系列和 VersaMax PLC 可能不支持

总线(BUS)指令

- BUS_RD_BYTE. 替换 90-70 系列 VME_RD_BYTE 功能

- BUS_RD_DWORD
- BUS_RD_WORD. 替换 90-70 系列 VME_RD_WORD 功能

注意: BUS_RD_ 指令替换 90-70 系列 VME_CFG_READ 功能

- BUS_RMW_BYTE. 替换 90-70 系列 VME_RMW_BYTE 功能
- BUS_RMW_DWORD
- BUS_RMW_WORD. 替换 90-70 系列 VME_RMW_WORD 功能
- BUS_TS_BYTE. 替换 90-70 系列 VME_TS_BYTE 功能
- BUS_TS_WORD. 替换 90-70 系列 VME_TS_WORD 功能
- BUS_WRT_BYTE. 替换 90-70 系列 VME_WRT_BYTE 功能
- BUS_WRT_DWORD
- BUS_WRT_WORD. 替换 90-70 系列 VME_WRT_WORD 功能

注意: BUS_WRT_指令替换 90-70 系列 VME_CFG_WRITE 功能.

新的跳变线圈和跳变结点

- NTCOIL
- NTCON
- PTCOIL
- PTCON

新的跳变结点 PTCON 和 NTCON 由相关的布尔变量上一次执行时的状态决定。存在的跳变结点 POSCON 和 NEGCON 由上一次写入与结点相关的布尔变量的值决定。

服务请求

PAC 系统服务请求（SVC_REQ）功能支持以下服务：

- #50: 读取逝去时间时钟 (2 个 DWORD)
- #51: 从扫描描开始时读取扫描时间 (DWORD)

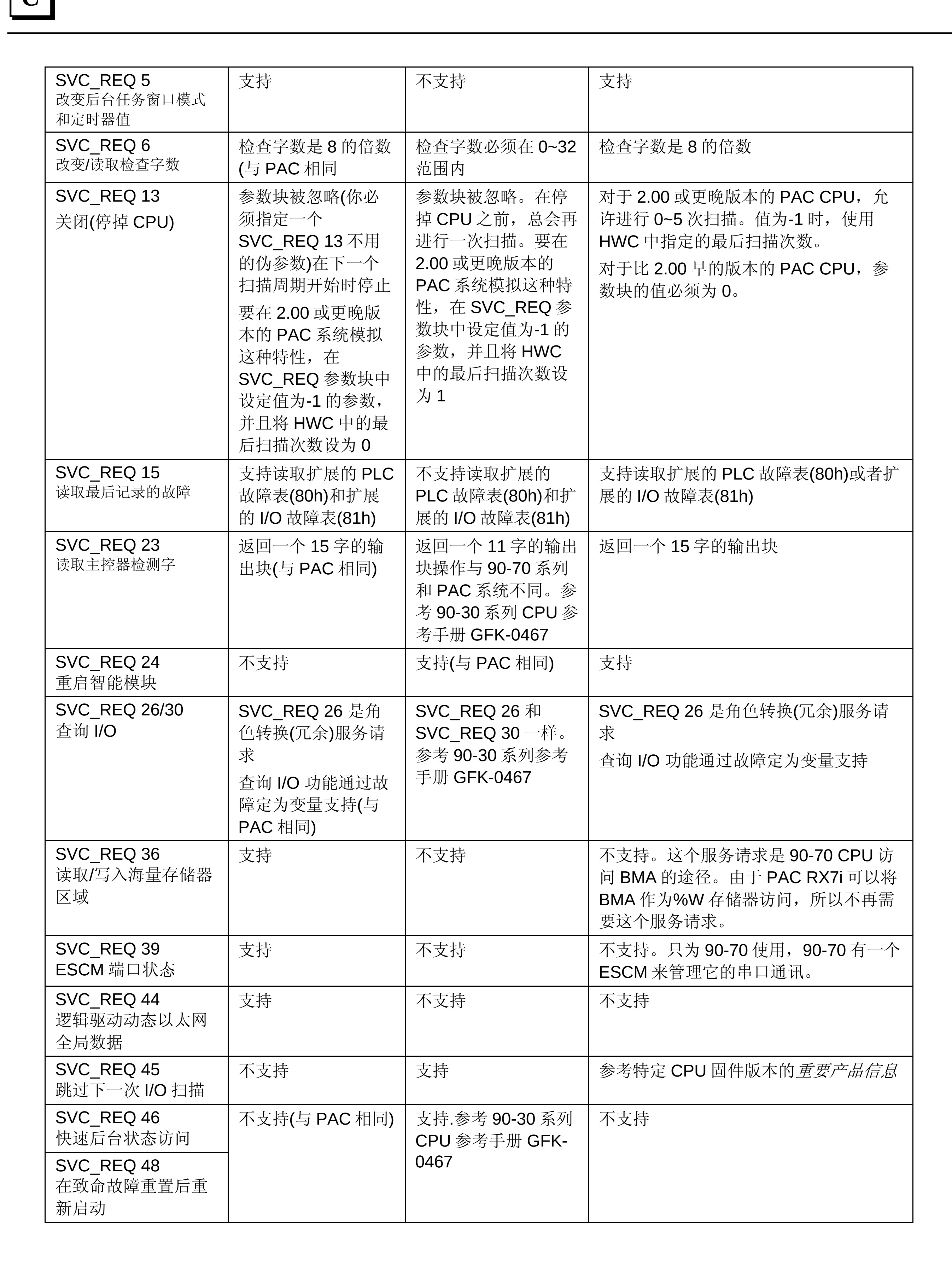
功能的差异

PAC 系统和 90 系列，VersaMax 等系统的不同大部分是因为 PAC 系统 CPU 支持以下新特征：

- 符号变量
- 非离散存储器中的位地址
- 足够长度的布尔数列可以替代其他数据类型的操作数。

下表列举了 90-70 系列和 PAC RX7i 内置功能的其他不同之处

种类	*90-70 系列*	*90-30 系列*	*PAC 系统*
ARRAY_MOVE_ (all mnemonics)	SR 参数不支持数据流	与 90-70 系列相同.	SR 参数支持数据流
ARRAY_RANGE_ (all mnemonics)	LL, UL, IN 和 Q 参数不支持数据流	不支持此功能	LL, UL, IN 和 Q 参数支持数据流. 操作数错误(变量超出范围)以及长度为 1 时，结果为 FALSE
BIT_SEQ	N (STEP) 参数不支持数据流	与 90-70 系列相同.	N 参数支持数据流.
位操作函数	如果多字功能的输入和输出变量地址范围重叠，会产生不可预知的结果。	与 90-70 系列相同.	输入和输出重叠时，PAC 系统使用功能激活时的数据，而不是先前执行时得到的输出数据。
COMM_REQ	支持 WAIT 模式 COMM_REQs.	与 90-70 系列相同.	不支持 WAIT 模式 COMM_REQs.
	如果你有 3 级或 4 级访问权限，智能模块可以自行更新状态字。	与 90-70 系列相同.	如果不包括 90-70 系列的 GBC，则在 2 级时可用。否则必须有 3 级或者 4 级权限。
	可在串口和以太网口上使用 COMM_REQ	与 90-70 系列相同.	可在串口和以太网口上使用 COMM_REQ。参考特定 CPU 固件版本的*重要产品信息*
	系统 ID 地址模块，占 2 槽	与 90-70 系列相同.	系统 ID 地址模块，占 1 槽(RX7i)
DATA_INIT_ASCII DATA_INIT_COMM DATA_INIT_DINT DATA_INIT_DWORD DATA_INIT_INT DATA_INIT_REAL DATA_INIT_UINT DATA_INIT_WORD DATA_INIT_DLAN	Q 输出参数不支持非直接变量和数据流	不支持此功能	Q 输出参数支持非直接变量和数据流
END	不支持	支持	不支持
Enhanced DO_IO	不支持	支持	不支持
RANGE_ (all mnemonics)	L1, L2 和 IN 参数不支持数据流	与 90-70 系列相同.	L1, L2 和 IN 参数不支持数据流



SVC_REQ 5 改变后台任务窗口模式和定时器值	支持	不支持	支持
SVC_REQ 6 改变/读取检查字数	检查字数是 8 的倍数(与 PAC 相同	检查字数必须在 0~32 范围内	检查字数是 8 的倍数
SVC_REQ 13 关闭(停掉 CPU)	参数块被忽略(你必须指定一个 SVC_REQ 13 不用的伪参数)在下一个扫描周期开始时停止 要在 2.00 或更晚版本的 PAC 系统模拟这种特性，在 SVC_REQ 参数块中设定值为-1 的参数，并且将 HWC 中的最后扫描次数设为 0	参数块被忽略。在停掉 CPU 之前，总会再进行一次扫描。要在 2.00 或更晚版本的 PAC 系统模拟这种特性，在 SVC_REQ 参数块中设定值为-1 的参数，并且将 HWC 中的最后扫描次数设为 1	对于 2.00 或更晚版本的 PAC CPU，允许进行 0~5 次扫描。值为-1 时，使用 HWC 中指定的最后扫描次数。 对于比 2.00 早的版本的 PAC CPU，参数块的值必须为 0。
SVC_REQ 15 读取最后记录的故障	支持读取扩展的 PLC 故障表(80h)和扩展的 I/O 故障表(81h)	不支持读取扩展的 PLC 故障表(80h)和扩展的 I/O 故障表(81h)	支持读取扩展的 PLC 故障表(80h)或者扩展的 I/O 故障表(81h)
SVC_REQ 23 读取主控器检测字	返回一个 15 字的输出块(与 PAC 相同)	返回一个 11 字的输出块操作与 90-70 系列和 PAC 系统不同。参考 90-30 系列 CPU 参考手册 GFK-0467	返回一个 15 字的输出块
SVC_REQ 24 重启智能模块	不支持	支持(与 PAC 相同)	支持
SVC_REQ 26/30 查询 I/O	SVC_REQ 26 是角色转换(冗余)服务请求 查询 I/O 功能通过故障定为变量支持(与 PAC 相同)	SVC_REQ 26 和 SVC_REQ 30 一样。参考 90-30 系列参考手册 GFK-0467	SVC_REQ 26 是角色转换(冗余)服务请求 查询 I/O 功能通过故障定为变量支持
SVC_REQ 36 读取/写入海量存储器区域	支持	不支持	不支持。这个服务请求是 90-70 CPU 访问 BMA 的途径。由于 PAC RX7i 可以将 BMA 作为%W 存储器访问，所以不再需要这个服务请求。
SVC_REQ 39 ESCM 端口状态	支持	不支持	不支持。只为 90-70 使用，90-70 有一个 ESCM 来管理它的串口通讯。
SVC_REQ 44 逻辑驱动动态以太网全局数据	支持	不支持	不支持
SVC_REQ 45 跳过下一次 I/O 扫描	不支持	支持	参考特定 CPU 固件版本的*重要产品信息*
SVC_REQ 46 快速后台状态访问	不支持(与 PAC 相同)	支持.参考 90-30 系列 CPU 参考手册 GFK-0467	不支持
SVC_REQ 48 在致命故障重置后重新启动			

SVC_REQ 49 自动重置统计表			
PID	逝去时间以 10 毫秒为单位计算	逝去时间以 10 毫秒为单位计算	修改了 PID 运算法则以去掉过去的一些错误情况，因此 PAC 的 PID 功能有些不同。 逝去时间以 10 微秒而不是 10 毫秒为单位计算。这是输出特性更加平滑，消除了余数累加到 10 毫秒所需的周期性的调节。 更多细节见第 10 章。
定时结点	从停止到运行转换时，重置定时结点的%S 变量 如果扫描时间大于定时结点时钟的一半，则强制转换。 停止模式时不更新	从停止到运行转换时，重置定时结点的%S 变量 如果扫描时间大于定时结点时钟的一半，则强制转换。 停止模式时不更新	基于自由运行定时器来确定每个定时结点的状态，这个定时器与每个扫描的开始没有关系。如果扫描时间与结点时钟保持一定的相位，则结点总会显示为同一种状态。例如，如果 CPU 为定常扫描模式，并且扫描周期设为 100 毫秒，则 T_10MS 和 T_100MS 位永远也不会触发。 停止模式时也更新
定时器功能	支持.程序块被调用是定时器功能开始累加时间	支持与 PAC 系统 和 90-70 系列操作不同。	与 90-70 系列相同。停止状态向运行状态转换时定时器功能开始累加时间 为防止定时器累加时间，可以使用程序块的 fst_exe 结点在程序块被调用后的第一次扫描时重置定时器。
VME_功能	支持	不支持	不支持。被 BUS_ 功能替换

浮点功能

和 90-70 系列, 90-30 系列以及 VersaMax CPUs 相比较，PAC 系统返回的非数字数据(NaN)可能略微不同。关于浮点功能返回的 NaN 数值，参考第七章“浮点数和操作错误”

浮点功能处理 NaN 的情况有所不同。PAC 系统 CPU 允许浮点硬件处理 NaN 案例，而不是将这些指令当作特殊情况来处理。影响：ADD, SUB, MUL, DIV, SIN, COS, TAN, ASIN, ACOS, ATAN, LOG, LN, EXP, EXPT, DEG_TO_RAD, RAD_TO_DEG, ABS_REAL, SQRT_REAL 函数

PAC 系统和其他 PLC 相比能计算更大范围的三角函数。结果是，之前返回 NAN 的某些值现在返回正常值。

在之前的 PLC 返回 NAN 的某些非正常状况下，目前的 PAC 会返回数值。特别是 0^0 返回 1 而不是 NaN

在线编辑模式

种类	***90-70 系列***	***90-30 系列***	***PAC 系统***
在线编辑模式和在线测试特征	不支持	不支持	对于那些允许在运行模式存储的逻辑变化，可以在线进行编辑和测试。关于这些特征的细节，参考编程软件在线帮助和 *Proficy™逻辑开发-PLC 入门* GFK-1918

变量

种类	*90-70 系列/90-30 系列*	*PAC 系统*
映射到离散存储器的多位变量	8-位和 16-位 变量	8-位, 16-位和 32-位 变量
	映射到 %I, %Q, %M, %T, and %G. 字变量某些时候可以映射到%S，依赖于指令	映射到 %I, %Q, %M, %T, and %G. 字变量某些时候可以映射到%S，双字变量某些时候可以映射到%SA, %SB, 和%SC，依赖于指令
非离散存储器中的位地址	不支持	你可以为 BYTE, WORD, INT, UINT, DINT 以及非离散存储器(%R, %AI, %AQ, %L, %P, and %W)中的 DWORD 变量的某一位确定地址
用于替换其它数据类型布尔数列	不支持	支持
非直接变量索引	16 位	指向%W 存储器时 32 位 指向其他存储器时 16 位
符号变量	不支持	符号变量时逻辑中没有分配地址的变量。Machine Edition 使用 PAC CPU%R, %AI, %AQ, %P, %L, %W, %I, %Q, %M, %T, %S 和%G 存储器之外的部分来处理所有的符号变量
公开变量	不支持	支持

系统变量

种类	*90-70 系列*	*90-30 系列*	*PAC 系统*
CPU 过热状态 (#OVR_TMP)	不支持	不支持	支持
故障定位系统变量	长度为 8 个字符	不支持	长度为 10 个字符
	可以定位 10 个槽#0 ~#9		可以定位 32 个槽#0 ~#31
	可以定位 32 个模块#0 ~#31		可以定位 256 个模块#0 ~#255

通讯

种类	***90-70 系列***	***90-30 系列***	***PAC 系统***
与 Machine Edition 通讯	猫, 串口和以太网	与 90-70 系列相同	任何版本都支持以太网。后续版本支持串口。参考特定 CPU 固件版本的*重要产品信息*
以太网适配器	以太网接口模块 IC697CMM742，任意机架	CPUs 364 and 374 有内置的以太网接口模块。以太网接口模块 IC693CMM321，任意机架	IC698CPE010, IC698CPE020 和 IC698CRE020 有一个内置的以太网子板。 以太网模块 IC698ETM001 只能放在 RX7i 主机架。最大数量：3 以太网模块 IC695ETM001 只能放在 RX3i 主机架。最大数量：4
配置以太网	包括 PLC 到你的计算机的串行电缆临时连接。	与 90-70 系列相同	可以为临时连接设定临时 IP 地址，这个过程中也可以设定永久 IP 地址。后续版本支持串口连接。参考特定 CPU 固件版本的*重要产品信息*
基于 Web 的数据监控	不支持	不支持	最多 16 个 web server 和 FTP 连接(加在一起 16 个)
网络路由	通过 CMM742 以太网接口配置来支持	不支持	不支持
串口	可用于与 Machine Edition 通讯 提供 SNP, Disabled 和 Custom 模式 参考串口通讯用户手册 GFK-0582.	与 90-70 系列相同	.端口 1 和端口 2 位外部设备提供串行接口。端口 1 也用于固件更新。板子上的第 3 个串口 用作以太网站管理器端口。 提供 RTU Slave, Message 和 SNP Slave, Serial I/O 和 Available 模式
缺省串口协议	SNP	SNP	Modbus RTU.
从 C 应用程序进行的串口通讯	Scanf 和 Printf.	与 90-70 系列相同	ANSI-类型读/写

EGD

种类	90-70 系列	90-30 系列	PAC 系统
EGD 变量	限制到 1,200		没有限制
EGD 上传	不支持从 PLC 向编程器上传 EGD 配置	与 90-70 系列相同	支持
广播 IP	不支持	不支持	广播选项为使用本地子网的广播地址进行 EGD 页的发送和接收提供了支持。
名称解析	支持	支持。与 90-70 系列限制相同。	不支持。EGD 页配置中的适配器名缺省为机架.槽号
接收周期	可配置	CPU364 可配置 CPU374 和 PAC 相同	为常量，200 毫秒，只能读取。
选择性接收	不支持	不支持	支持. EGD 接收页的范围可设定为忽略
%W	N/A	N/A	在 EGD 页配置中支持

闪存

种类	90-70 系列	90-30 系列	PAC 系统
清除闪存	不支持	与 90-70 系列相同	支持. 清除操作会影响所有项目(HWC, 逻辑和初始的/强制的数值).
读取/校验逻辑	可以分别读取/校验逻辑和 HWC	与 90-70 系列相同	不可以分别读取/校验逻辑和 HWC
如果有 EGD	不可以读取，写入，校验闪存	可以读取，写入，校验或者清除闪存(与 PAC 相同)	可以读取，写入，校验或者清除闪存
从闪存中重新存储	N/A	N/A	OEM 商可以将 PAC 系统配置为自动从闪存向电池为后备电源的 RAM。进行这种存储后，只要 RAM 正常，PLC 就从 RAM 启动，而不是从闪存启动(和上电检测时一样)
向闪存写入	不写入跳变	不写入跳变	对当前设定并写入闪存的跳变进行快照
校验闪存	不校验跳变	不校验跳变	校验跳变。如果跳变从 0 变到 1 在变回 0，数值相等，但是跳变不等。
从闪存中存储/重新存储跳变位	当把变量表从闪存时，会在重新存储时根据 RAM 和闪存内容的不同来设定跳变位。		向闪存/从闪存中存储/重新存储跳变位和状态值时，会将变量表中的每一位覆盖
逻辑/配置源和闪存中没有配置时的 CPU 模式以及上电来源是 RAM 的情况下为总选择闪存(Always Flash)		使用缺省逻辑/配置并且进入停止/不进行 IO 扫描(Stop Disabled)模式	使用 RAM 中的逻辑/配置。CPU 模式按第五章"上电时的逻辑/配置对 CPU 特性的影响"

可中断的闪存读/写	必须在可以取消之前完成整个传输过程	必须在可以取消之前完成整个传输过程	闪存或者 RAM 中的内容作为一个独立的文件拷贝。这使你可以在拷贝过程中取消闪存读写，而不用非等到整个传输过程完成。

存储器

支持的存储器区域的不同

存储器区域	*90-70 系列*	*90-30 系列*	*PAC 系统*
%GA~%GE	支持	使用%G (与 PAC 相同).	使用%G 将 90-70 PLC 的对象转换到 PAC 时，自动将%GA - %GE 存储器映射到%G。详见 C-24 页“转换过程中的变化”
海量存储器区域	通过 SVC_REQ 36 访问	N/A	将变量映射到%W 存储器
%W	在 LD 程序中，通过 SVCREQ 36 功能访问 BMA。在 C 程序中，通过 PLCC_buil_mem()功能访问 BMA。	N/A	支持。替代海量存储器区域
符号变量	不支持	与 90-70 系列相同	支持。一个存储器区域用于离散符号变量。另一个存储器区域用于非离散符号变量。

最大存储量

PAC 对象一些存储器区域的存储量大小要比 90 系列 PLC 对象的存储器区域的存储量要大。

存储器区域	*90-70 系列*	*90-30 系列*	*PAC 系统*
%AI, %AQ	8,192 字(16,384 字节)	32640	每一个 32,640 字(65,280 字节)
%I, %M, %Q	12,288 点 (1,536 字节)	2048	每一个 32,768 点(4,096 字节)
%T	256 点(32 字节)	256	1,024 点(128 字节)
%R	16,384 字(32,768 字节)	32640	32,640 字(65280 字节)
%W	不支持	不支持	用户可用的 RAM 的最大值
总计用户空间			10 兆字节

关于如何计算用户存储器，参考附录 B.

关于存储器的其他不同

存储器区域	*90-70 系列*	*90-30 系列*	*PAC 系统*
%P 和%L	缓存大小不可配置	不支持	缓存大小可以通过 LD 块的额外本地变量属性配置

冗余

种类	***90-70 系列***	***90-30 系列/RX3i***	***PAC 系统***
CPU 冗余	支持	不支持	冗余 CPU 支持
Genius 冗余	支持		支持 冗余 CPU 支持 Genius 上的 CPU
IP 地址冗余	支持		冗余 CPU 支持

Genius

种类	***90-70 系列***	***90-30 系列***	***RX3i***	***PAC 系统 RX7i***
处理 GBC 丢失	不管为每个设备设定的缺省参数值为多少，都为与丢失的 GBC 相关的设备设定输入数据以保持最后一次扫描时的状态。	将与丢失的 GBC 相关的设备的输入数据设为 0。	将与丢失的 GBC 相关的设备的输入数据设定为 GBC 参数缺省值。(缺省/保持最后一次扫描时的状态。)	将与丢失的 GBC 相关的设备的输入数据设定为 GBC 参数缺省值。
Genius 设备丢失和增加情况的处理	90-70 系列 GBC 记录丢失设备 (Loss of Device)或者增加设备(Addition of Device)故障，90-70 系列 PLC 更新对应的故障定位变量。	90-30 系列 GBC 不记录故障。90-30 系列 PLC 不支持故障定位变量。	90-30 系列 GBC 不记录故障，因此 RX3i 不更新故障定位变量。	90-70 系列 GBC 记录丢失设备 (Loss of Device)或者增加设备(Addition of Device)故障，因此 RX7i 更新对应的故障定位变量。
冗余 Genius 模块丢失时的数据处理	立即用缺省输入数据替换输入变量表，并且更新相关的自诊断信息。	N/A	NA	在发生丢失设备后的第一次输入扫描过程中用缺省数据更新输入数据和自诊断表，并且更新相关的自诊断信息。在发生丢失设备后的第一次输出扫描过程中更新输出自诊断数据表。
强制当前状态位的设定 FRC_PRE %S12.	限定为 Genius 模块	NA	不限定为 Genius 模块。任何输入模块报告出现强制，都会将这一位置位	不限定为 Genius 模块。任何输入模块报告出现强制，都会将这一位置位
点故障变量	任何 90-70 系列 CPU，90-70 系列 GBC 和其他遗留的 90-70 系列模块都支持。	任何 90-30 系列 CPU 和其他遗留的 90-30 系列模块都不支持。	RX3i CPU 支持点故障变量。90-30 系列 /RX3i GBC 不支持	支持

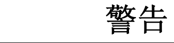


I/O 和智能模块

种类	*90-70 系列*	*90-30 系列*	*PAC 系统*
离散输出模块	90-70 系列 CPU 控制的离散输出模块在第二次扫描后驱动输出。 因为在成为输出扫描的一部分之前，DO I/O 不会被使能，所以在第一次扫描时 DO I/O 不会有影响。	与 PAC 系统相同	在成为输出扫描的一部分时，PAC 系统 CPU 控制的离散量输出模块将输出量设定为用户逻辑中所确定的值 DO I/O 功能其执行时会触发模块输出（如果模块的输出没有被使能）
与高速计数器进行 I/O	如果指定%AI 则返回 AI 输入数据。如果指定%I 则返回所有输入数据((I and AI))	与 PAC 系统相同	返回所有的输入数据(I 和 AI)
I/O 中断块	定时中断优先级别大于 I/O 中断	NA (不支持 I/O 中断)	按先来先处理的原则编列顺序。 后续固件版本的 CPU 允许进行优先权编制，对于特定版本的 CPU 参考随 CPU 附带的 IPI 文件
为不可配置和/或不可启动(enabled)中断的 I/O 地址安装一个 I/O 中断块	无故障 – CPU 可以进入运行模式	NA (不支持 I/O 中断)	在下装到 CPU 后，CPU 转到停止-故障状态。要想使 CPU 回到运行状态，必须清空 PLC 故障表。 必须在硬件配置中将模块中断设为 configured/enabled
高速计数器和 IC697MDL641 模块上的 I/O 中断	没设定继承跳变(POSCON/NEGCON)	N/A	中断发生时，每一个离散变量都要设定继承跳变(如果某一位的新状态为 ON，则他的上升沿结点为 TRUE。如果新状态为 OFF，则他的下降沿结点为 TRUE。)
模拟量输入中断	中断逻辑运行时不设定 HIALR 和 LOALR 故障结点。	不支持点故障	如果点故障使能，则在运行中断逻辑之前将 HIALM 或者 LOALM 故障结点设为 TRUE

模拟量扩展模块	对扩展器的数目没有限制。 模拟基板和扩展模块可以使用不同的扫描设定。	N/A	RX7i 机架上最多可以用 3 个扩展器(只适用于 RX7i) 扩展模块和模拟基板必须有同样的扫描设定。

将 90-70 系列应用程序转化为 PAC 应用程序



将 **90-70** 系列应用程序转化为 **PAC** 应用程序时，程序的执行结果可能会不同。程序开发人员有责任在将程序投入到生产环境之前校验和测试应用程序的执行情况。

注意

对象转换时不能回转的。在转换过程中将逻辑块删除以后就不能重新存储回来。即，不能撤销转换。建议在转换对象之前将工程备份。

注意: 如果 90-70 系列对象包含状态逻辑，则不能将此对象转换为 PAC 对象

转换准备

扩展器模块

如果你要转换的 90-70 系列机架配置中包含 IC697ALG440 或者 IC697ALG441 模拟量输入扩展器模块，你必须确定 IC697ALG230 基块和扩展器块按如下规则放置在指定的槽位：否则这个模块将不会转换。(如果没有使用扩展器模块，基块可以放置在任何槽位)。如果你使用扩展器模块，你必须在转换之前将基块放置在 90-70 机架的 3，4，5 槽上

- 如果你将基块放置在 90-70 机架的 3 槽上，你必须将 IC697ALG440 或者 IC697ALG441 扩展器模块放置在 4，5，6 槽上
- 如果你将基块放置 4 槽上，你必须将扩展器模块放置在 5，6 槽上
- 如果你将基块放置 5 槽上，你只能有一个扩展器模块，且必须将其放置在第 6 槽上

注意: 另一个可转换的配置是一个基块放置在第 3 槽，他的扩展器模块放置在第 4 槽，另一个基块放置在第 5 槽，他的扩展器模块放置在第 6 槽。

PCM, CMM 和 DLAN 模块

如果你要转换的 90-70 系列机架配置中包含 IC697PCM711, IC697CMM711 或者 IC697BEM761 模块，你必须确定这些模块在 RX7i 机架中的槽位号不能大于 9。如果这些模块在 90-70 机架中的槽位号大于 5，将不能转换，因为对应的 RX7i 机架中的槽位号大于 9。例如，IC697PCM711 模块在 90-70 机架中的槽位号为 6，则不能转换，因为对应的 RX7i 机架中的槽位号为 11。

注意: 如果你的 90-70 应用程序中使用了 PCM, CMM 或者 DLAN 模块, 关于 PAC RX7i 模块使用方面的信息，请参考第十一章“90-70 系列通讯和智能模块”。

VME_ 指令

PAC RX7i 提供的 Bus_Read 和 Bus_Write 功能与 90-70 系列使用的 VME_Read 和 VME_Write 功能类似。转换 90-70 系列应用程序时，对于硬件配置和逻辑的更改需要使用新的指令。这些改变必须在转换应用程序后手动进行。在转换应用程序之前，回顾一下有关 VME 地址的描述以及完成转换所需的信息。

注意: 关于在 PAC 系统中选择，配置和编程非 GE Fanuc VME 模块的详细信息，参考 *PAC RX7i VME 模块集成用户指南* GFK-2235.

PAC 系统 vs. 90-70 系列 VME 地址编制

非 GE Fanuc VME 模块

典型的 VME 模块配置为对应特定的 VME 地址。多数的非 GE Fanuc 板子本身对应基地址，并且使用一个或几个跳线来设置偏移地址(AM 码)。典型情况是一个板子可以访问一个或者两个特定的存储器区域。PAC 系统将这些区域认定为 *Regions*。一个 Region 由可访问的基地址，AM 码，大小或者宽度，终端接口类型(byte, word, dword 等等)组成。

PAC 系统中，在硬件配置中指定每个 VME *Region*。PAC 系统中每个模块最多可以有 8 个 *Region*。每一个 Region 包括 *Region 号，*基地址，AM 码，大小和接口类型。关于这些设定的说明参考编程软件帮助信息。PAC 系统 BUS_功能所包含的 region (RGN)参数在硬件配置信息中配置。

90-70 系列中，AM 码和 VME 地址作为 VME_功能自身的参数。宽度(width)由使用的指令确定(例如，READ_BYTE 与 READ_WORD)



完成转换所需的信息

要完成转换进程，你必须确定每个 VME 模块所使用的存储器区域，包括 AM 码和 VME 基地址。这个信息随使用的 VME 板提供。如果不可用，可以参考 90-70 系列应用程序逻辑中所包含的信息。作为一个通用规则，每个 AM 码都要使用单独的 REGION

PCM 应用程序

PAC 系统为 VME 模块分配的 VME 地址和 90-70 分配的 VME 地址不同。如果 IC697PCM711 上运行的应用程序访问 VME 总线(即，使用 Set_vme_ctl, Vme_read, Vme_test_and_set 或者 Vme_write)，则这个程序使用的 VME 地址必须更新为满足 PAC RX7i VME 地址分配规则。要确定 RX7i 上所使用的 VME 地址正确，请参考 PAC RX7i VME 模块集成用户指南 GFK-2235 的下列部分：

“主机架 GE Fanuc 模块 VME 地址”
“扩展机架 GE Fanuc 模块 VME 地址”

同时也请注意 Vme.h 中的 PCM C 工具提供的下列 S9070_xxxx 宏不能用于计算 RX7i 系统中的 VME 地址

S9070_RACKSLOT_VALID(r,s) (r>=0&&r<=7&&s>=2&&s<=9)

S9070_VME_HI_ADDR(r,s) ((r)?(0xF0-(0x10*(r))+2*((s)-2)):(2*((s)-2)))

S9070_VME_SHORT_ADDR(s) (0x800*s)

对象转换

在编程软件中将 90-70 系列对象转化为 PAC RX7i 对象：

1. 在浏览器的工程键下，右键单击想要转换的对象并且选择属性。在 Inspector 中显示对象属性。
2. 在属性中，选择 PAC RX7i 家族。会显示转换警告信息。如果你想继续转换，点击 OK。
3. 对象转换为 PAC RX7i 家族。转换完成后，InfoViewer 中会显示转换报告。
4. 浏览转换报告，并且纠正发现的问题，然后校验应用程序。在将程序应用到生产环境之前，必须彻底的测试和检测由于执行差异造成的问题。

   更多信息请参考 C-24 页“转换造成的差异”和 C-25 页“完成转换”



将 90-70 系列应用程序转化为 PAC RX7i 应用程序时，执行结果可能与原来程序有差异。程序开发人员有责任在将程序投入到生产环境之前校验和测试应用程序的执行情况。

将 90-70 系列应用程序转化为 PAC RX7i 应用程序时的变化

PAC RX7i 对象会将支持的每一个 90-70 系列模块从 90-70 机架的双宽度槽转换为对应 RX7i 机架上的单宽度槽上。

注意: 因为 PAC 系统 RX7i 的电源只是用一个槽，所以 PAC 系统 RX7i 的槽号如下计算：

PAC 系统 RX7i 槽号 = (2 * [90-70 系列槽号]) - 1

- 尽可能保留每个转换模块的参数值。只适用于 RX7i 的参数设定为缺省值。
- 以太网全局数据(EGD)转换。对于每个机架，适配器 1 转换为 RX7i CPU 内置以太网子板，然而适配器 2，3，4 转换为 RX7i 以太网模块(IC698ETM001)，适配器 5 或 5 以上被忽略。

注意: 90-70 系列适配器 1 所在槽号在 RX7i 中是空的，RX7i 空槽=(2 * [90-70 系列适配器 1 的槽号]) - 1

- %GA - %GE 变量存储区域按一下缺省规则转换为%G 地址：

转换前的存储器类型	*转换后的存储器类型和偏移量*
%G	%G+0
%GA	%G+1280
%GB	%G+2560
%GC	%G+3840
%GD	%G+5120
%GE	%G+6400

- C 块会保留下来并且直接在报告中标注。你可能需要编辑 C 块。你也需要用 PAC 系统 C 工具包重新编译这些 C 块并且在 PAC 对象中更新
- 所有的 C 程序被删除。
- LD 块被转换并且检查需要更新的指令。

注意: 转换后的对象的第一次校验标示出 RX7i 不支持的 LD 指令

- 除了用于逻辑的以外， 90-70 系列原始的系统变量被删除。PAC RX7i 的系统变量增加到对象当中。逻辑中使用的每个系统变量都会在报告中作出警告。这提示你确定这些系统变量在新系统中是否还正确。
- 90-70 系列对象转化为 RX7i 对象时，所有的 90-70 系列故障定位系统变量转化为 RX7i 版本的变量，在槽号之前加了一个 0。例如，如果#BUS_121 用于 90-70 系列对象，则在转化为 RX7i 后，变量名称改为#BUS_1021。同时，#RACK_0r 转化为#RACK_00r。
-

完成转换

浏览转换报告

转换完成后的转换报告概括了硬件配置转换和逻辑转换的结果。标明了没转换的项目。

注意: 转换报告不会对所有可能的逻辑执行差异都作出警告。转换完成后的正确性校验可能会报告出一些转换过程中没有发现的问题。即是转换报告和正确性校验过程中都没提及，90-70 系列到 PACRX7i 转换时也可能会产生一些执行差异。

报告提供了一份对每个 LD 块的分析并且对不支持的指令，不支持的服务请求，故障定位变量使用作出报告。另外还报告了转换的指令以及不能转换的指令和为什麽不能转换。

用红色字符对逻辑执行过程中大部分潜在的重大差异作出了警告。(例如，由于 90-70 系列和 PACRX7i 执行顺序的不同造成的逻辑分歧)对于每个潜在差异的报告，你应该检查逻辑。查找有差异的指令中的输入输出变量，尤其是输入和输出量不在同一行的情况。

报告保存在 PAC RX7i 对象附带文件夹的 Documentation 文件夹中。你可以从 InfoViewer 直接打印报告，也可以从 Documentation 文件夹打印报告

用 BUS_ 指令替代 VME_ 指令

RX7i 提供了一套可以用于访问总线上模块的数据的 BUS_指令：BUS_RD, BUS_WRT, BUS_RMW 和 BUS_TS.这些指令类似于 90-70 系列支持的 VME_ 指令。90-70 系列的 VME_ 指令不会自动转换为 RX7i 的 BUS_指令。

下面概括了将应用程序从 90-70 系列转换到 PAC 所要做的改变：

- 确定特定子板所需要的 region 的数目，每个 regions 的 AM 码和基地址。注意 AM 码需要度单独的区。
- 使用 Machine Edition 硬件配置来配置每个非 GE Fanuc 板，增加合适的 region 数，表明需要的 AM 码，基地址，大小和接口类型。每个 region 根据 LD 程序内的数字确定。
- 更改 LD 功能调用：
  - 用 BUS_指令替换掉对应的 VME_指令。(例如,用 BUS_READ_WORD 替换掉 VME_READ_WORD；用 BUS_READ_指令替换 VME_CFG_READ 等等) 不能替换的指令在对象转换报告中标明。
  - 为非 GE Fanuc VME 模块增加正确的机架号和槽号
  - 在硬件配置中标明正确的 Region 号。
  - 计算功能的 “地址”参数。注意这个值时对应 VME 绝对地址的偏移地址。因此，你必须用想要设定的地址减去基地址。例如，如果 90-70 系列指令使用地址 0x400100，则你需要将 region 的基地址设为 0x400000，将偏移地址设为 0x100
  - 如果你的 RX7i 应用程序需要访问 PCM，CMM 或者 DLAN 的双端口寄存器，可以使用 BUS READ 和 WRITE 功能。访问这些模块中的一个时，将功能的 Region 参数设为 1。(对于 PCM，CMM 和 DLAN，region 1 被预设定为模块的整个双端口寄存器。根据这些模块的目录号培植这些模块；不要将这些模块作为 VME 模块配置)

注意还需要配置其他(可选择的)参数。可以在 Machine Edition 在线帮助中找到这些参数的配置方法。

为从 90-70 系列转换过来的程序增加堆栈分配

90-70 系列程序转化为 RX7i 程序时使用相同的堆栈分配。RX7i 比 90-70 系列使用更多的堆栈空间，所以转换后一些用户程序可能不再运行。要增加堆栈空间，右键单击_MAIN 块，选择属性项。堆栈大小(Stack Size)在属性页的最后一栏。RX7i 缺省的堆栈大小为 64KB。具有更大数量的嵌套调用的程序可能需要更多的堆栈空间。一般来讲，由 90-70 系列转化而来的 RX7i 程序所需的堆栈空间应为原来程序所需堆栈空间的 3 倍左右。如果程序使用的堆栈空间超出限制，会显示一个自诊断故障。

将 90-30 系列应用程序转化为 PAC 应用程序

转换准备

End 指令

PAC 系统不支持 END 指令，END 指令无条件终结程序执行，并开始下一次扫描(从_MAIN 块的第一行开始执行)。下列对使用 END 指令的程序重新编制的建议：

- 增加 JUMP(s)，并将 JUMP(s)指向调用链的程序块的末端
- 将 END 指令后的程序做到一个单独的程序块内，并且只有在不执行 END 指令时才执行这个块。
- 用其他方式调试 LD 程序，例如当调试环境满足时将临时寄存器条目保存。

更新 24V 模拟量模块 (IC693ALG220, IC693ALG221, and IC693ALG222C)

如果你使用这些模块的老版本(ALG220G, ALG221G, ALG222C 或更早的)，你必须为背板终端提供电源或者用新版本模块替换掉老版本模块。

与早期用户的 SRTP 通讯

RX3i CPU 可以放在 0 号机架的任何一槽内(除了最后一槽)。PAC 系统提供了 SRTP 目的地服务，这个服务允许外部用户使用 SRTP 寻找 CPU 所在机架以及 CPU 的槽号，以便于 CPU 进行通讯。然而，先前的用户，比如使用 SRTP 通道和 HCT 主机的 90 系列 PLC 应用程序不能使用这个服务。他们假定 CPU 总是和 90 系列系统一样在 0 机架第 1 槽。

为了支持使用 SRTP 通道的 90 系列 PLC 等用户，RX3i 会将 SRTP 服务器连接传来的服务请求从 0 机架第 1 槽导到 0 机架第 2 槽。在如下情况时，会进行导向操作：

- 系统中只有一个 CPU
- CPU 放置在 0 机架第 2 槽
- 远端用户在连接并且查找机架号和槽号时没有使用 SRTP 定位服务。

将服务从 0 机架第 1 槽导到 0 机架第 2 槽只是将服务请求信息的目标地址中的机架和槽位端口号更改一下。服务请求的内容不会检查或修改，从 CPU 传回来的服务请求相应信息也不会更改。

90 系列 SRTP 通道用户所使用的所有服务和所有的 HCT 服务都可以成功的导向。

CPU 槽位

RX3i CPU (IC695CPU310)是一个双槽宽度的模块，装在机架上时连接器连接的是右侧的槽位。用户逻辑应用程序使用和进行配置时以它所占据的最左槽位判断。例如，如果 RX3i CPU 物理连接器插入到第 4 槽，这意味着它占用第 3，4 槽，此时认为 CPU 放置在第 3 槽。CPU 位置不是以他的连接器连接的槽位来判断位置，而是以整个 CPU 模块占用的最左侧槽位来定位 CPU.

机架为 12 槽时，RX3i CPU 可以放在出 11 槽之外的任何一槽内。机架为 16 槽时，RX3i CPU 可以放在出 15 槽之外的任何一槽内。上述机架最后一槽是给扩展模块预留的，CPU 占两个槽位，放置在倒数第二槽时连接器会连接到预留槽位上，所以不能将 CPU 放置在该槽上。

将 90-30 换为 PACSystems RX3i CPU 时，注意第一槽放置 CPU，电源模块只占用 1 个槽位时才能在第 0 槽放置电源，。因此，如果应用程序使用已经存在的 90-30 系统，就必须把第一槽留给 CPU，并将占用两个槽位的电源模块放置在第 1 槽的 RX3i CPU 的右边

决定不是将 CPU 放置第 1 槽，而是将其放置在其他槽位时，需要注意转移应用程序时可能出现的问题。下表列举了更换 CPU 槽位时对程序转移可能造成的影响。

CPU 槽位放置问题

影响的项目		*如何影响*
用户逻辑	服务请求 #15 (读取最后记录的故障)	CPU 故障不是标准的 0.1 位置，而是反映 CPU 所在槽位。用于故障表解码的用户逻辑可能需要更新服务请求才能正常运行。
	服务请求 #20 (读取故障表)	
	通讯请求 (COMM_REQ)	与 CPU 之间的 COMM_REQs 需要用正确的 CPU 槽变量更新。
H/W 配置	CPU 槽位	硬件配置中的 CPU 位置必须更新，以反映真实的 CPU 位置
故障表	记录的 CPU 故障	CPU 故障不是标准的 0.1 位置(机架号.槽号)，而是反映 CPU 所在槽位。
外部设备	*90 系列 PLC*	
	使用 SRTP 通道 COMM_REQs 的远端 90 系列 PLC 假定 CPU 在第 1 槽。为了支持与 90 系列 SRTP 用户(比如使用 SRTP 通道的 90 系列 PLC)进行通讯，如果 CPU 放置在 0 机架，第 2 槽(远端用户不会发出 SRTP 目的地请求来确认 CPU 所在的机架和槽号)，则 RX3i 内部将收到的 SRTP 请求从{ 0 机架, 第 1 槽}转到{ 0 机架, 第 2 槽}。这个特殊的转向功能使原来假定电源放在 CPU 左侧的 90-30 应用程序能够正常工作。从 90 系列 PLC 和非第 1，2 槽放置的 CPU 建立通道的尝试会失败。	
	HMI 和外部通讯设备	
	所有和 CPU 相互作用的外部通讯设备会检查与不放在第 1 槽的 CPU 进行兼容性检查。可能会出现问题，但不会限制初始化连接顺序和故障报告。用户应选择 Machine Edition 中的“GE SRTP”作为通讯驱动器，它允许与放置在任何槽的 CPU 进行通讯。	
	主机通讯工具 (HCT)	
	使用主机通讯工具的应用程序可能要求更新驱动器。	

对象转换

在编程软件中将 90-30 系列对象转换为 PACSystems RX3i 对象的步骤：

1. 在浏览器的工程键下，右键单击想要转换的对象并且选择属性。在 Inspector 中显示对象属性。
2. 在属性中，选择 PAC RX3i 家族。在 InfoViewer 中会显示转换警告信息。如果你想继续转换，点击 OK。



3. 对象转换为 PAC RX7i 家族。转换完成后，InfoViewer 中会显示转换报告。
4. 浏览转换报告，并且纠正发现的问题，然后校验应用程序。在将程序应用到生产环境之前，必须彻底的测试和检测由于执行差异造成的问题
5. 更多信息请参考 C-30 页“转换造成的差异”和 C-31 页“完成转换”

Warning

将 90-30 系列应用程序转化为 PAC RX3i 应用程序时，执行结果可能与原来程序有差异。程序开发人员有责任在将程序投入到生产环境之前校验和测试应用程序的执行情况。

将 90-30 系列应用程序转化为 PAC RX3i 应用程序时的变化

硬件配置

- PAC RX3i 对象会将支持的每一个 90-30 系列模块从 90-30 机架和槽号转换为对应 RX3i 机架和槽号。
- 因为 RX3i CPU 和缺省电源需要用两个槽，其他模块根据需要调整。
- 带内置以太网子接口的 90-30 系列 CPU 转换为以太网全局数据(EGD)转换。对于每个机架，适配器 1 转换为 RX3i CPU 和 IC695ETM001 外围以太网模块。槽位其他模块根据需要调整。
- 尽可能保留每个转换模块的参数值。只适用于 RX3i 的参数设定为缺省值
- 电源消耗要求从瓦转为安培

逻辑

- C 块会保留下来并且直接在报告中标注。你可能需要编辑 C 块。你也需要用 PAC 系统 C 工具包重新编译这些 C 块并且在 PAC 对象中更新
- 所有的 C 程序被删除
- IL (指令表)和 SFC (顺序功能表) 不会转换。 不支持 IL 和 SFC 程序
- LD 块被转换并且检查需要更新的指令。

不支持下列指令：

- SVC_REQ #41 (PEEK), SVC_REQ #42 (子板信息).这些不会转换。提供了其他调试系统和确定子板修改信息操作的方式。
- SVC_REQ 46 快速背板状态访问
- SCV_REQ #48 & #49 (自动重启参数).以为不执行自动重启特征，所以不翻译。成功编译的程序不包括这些东西，已经将其忽略掉。
- END. 不支持

下列功能需要手动转换：

- SVC_REQ 6,改变/读取检查字数目
- SVC_REQ 15,读取最后一次记录的故障
- SVC_REQ 23,读取主控器检查字
- SVC_REQ 26/30,查询 I/O。注意 PACSystems 通过故障定位变量支持 90-30 系列的查询 I/O 功能。

T 下列指令有所改变：

- WORD_TO_REAL 指令转换为 UINT_TO_REAL.
- REAL_TO_WORD 指令转换为 REAL_TO_UINT.
- 增强型 DO_IO 转换为标准 DO_IO (不再使用为常量的 ALT 参数，忽略该参数)
- 非嵌套 JUMP, LABEL, MCR, & ENDMCR. 这些转换为对应的嵌套 JUMPs, LABELs, MCRs, & ENDMCRs.

完成转换

浏览转换报告

转换完成后的转换报告概括了硬件配置转换和逻辑转换的结果。标明了没转换的项目。

注意: 转换报告不会对所有可能的逻辑执行差异都作出警告。转换完成后的正确性校验可能会报告出一些转换过程中没有发现的问题。即是转换报告和正确性校验过程中都没提及，90-30 系列到 PAC 转换时也可能会产生一些执行差异。

报告提供了一份对每个 LD 块的分析并且对不支持的指令，不支持的服务请求，故障定位变量使用作出报告。另外还报告了转换的指令以及不能转换的指令和为什麽不能转换。

用红色字符对逻辑执行过程中大部分潜在的重大差异作出了警告。对于每个潜在差异的报告，你应该检查逻辑。查找有差异的指令中的输入输出变量，尤其是输入和输出量不在同一行的情况。

报告保存在 PAC 对象附带文件夹的 Documentation 文件夹中。你可以从 InfoViewer 直接打印报告，也可以从 Documentation 文件夹打印报告